AD-A063 702 BURROUGHS CORP PAOLI PA FEDERAL AND SPECIAL SYSTEMS GROUP F/G 17/2
EXPLORATORY SYSTEM CONTROL MODEL DEVELOPMENT (ESMD). VOLUME I. --ETC(U)
JAN 78 DCA100-76-C-0081
UNCLASSIFIED 64297-VOL-1 SBIE-AD-E100 144 NL

1 OF 2
AD
A063702

Volume I
Sections 1-3

64297
January 1978

FINAL REPORT

FOR THE EXPLORATORY SYSTEM CONTROL MODEL DEVELOPMENT (ESMD)

for

THE DEFENSE COMMUNICATIONS AGENCY
WASHINGTON, D.C. 20305

Burroughs Corporation
Federal and Special Systems Group
Paoli, Pa. 19301

Volume I

Sections 1-3

64297

January 1978

See back page for 1473

Burroughs

FINAL REPORT

FOR THE EXPLORATORY SYSTEM CONTROL MODEL DEVELOPMENT (ESMD)

for

THE DEFENSE COMMUNICATIONS AGENCY
WASHINGTON, D.C. 20305

Burroughs Corporation
Federal and Special Systems Group
Paoli, Pa. 19301

Table of Contents

Pages 2-35 and 2-36 are left out intentionally because they contain proprietary information

List of Figures

1. Modeling Capability for System Control Distributed Data Base

1.1 Data Base Structures and Order Wire Schemes

The discussion that follows and Appendix A comprise task 1.1. Appendix A covers the data base structures portion of the task. Order wire schemes are discussed below in terms of the five loop (the model) system and in terms of much larger systems. The order wire modes discussed are CRT-to-CRT attachment and CRT-directed data-base transfer commands.

1.1.1 Order Wire in the Model

In the model, the direct CRT-to-CRT mode is implemented through an ATTACH command. The ATTACH command may be entered at any loop-connected CRT in the form "ATTACH nn" where nn is the logical ID of the receiving CRT. The allowable CRT's in the model have the logical ID's (node designators) 4, 8, 18 and (for loop 5) 25. When this is done, the CRT is no longer connected to a dialog director. Its messages go directly to the CRT addressed. The recipient CRT is still under the control of its dialog director. Its return messages to the sending CRT may be sent through the use of mode 1 of the user language or an ATTACH command may be used to supply direct two-way conversation.

A DETACH command may be used on CRT's with logical ID's 18 and 25 but not on those with logical ID's 4 and 8. The easiest way to detach 4 and 8 is to clear the loop to which it is attached. An alternate (though awkward) method is to supply suitable mode 3 commands from another CRT. For the commands in mode 3, see the User Manual for the ESM.

When both CRT's are attached, a direct conversation can take place without the use of a directing host computer.

For file transfers only (data-base transfers are not implemented at present), the following method is suggested.

(a) Use an ATTACH command to the host that is to receive the file if your CRT is not already attached. The host logical ID's are 1, 5, 16 and (for loop 5) 24.

(b) Perform an ABORT command to return the host to its basic operating state and then run the loop file transfer utility (LPFT) program.

(c) DETACH (or clear) and repeat (a) and (b) for the sending host.

(d) Use LPFT as described in ESMD documentation to achieve the required file transfer.

1.1.2 Large System Partitioning

In large systems where more than 250 logical ID's (or 500 or 1000 depending on the size of the address field) are used, it becomes convenient to partition systems into subsystems. One type of hierarchical general approach will be given and this will be applied to a System Control architecture.

Consider a system as shown in figure 1.1-1 showing four subsystems each containing up to 250 units. Each unit within a subsystem has its own unique logical ID (LID). Any messages within the subsystem requires only a single byte per LID. Each subsystem has gateways to other subsystems which are called "ports". Each port has an LID within its own subsystem. The message header would show (among other things) the receiving LID followed by the sending LID in the form

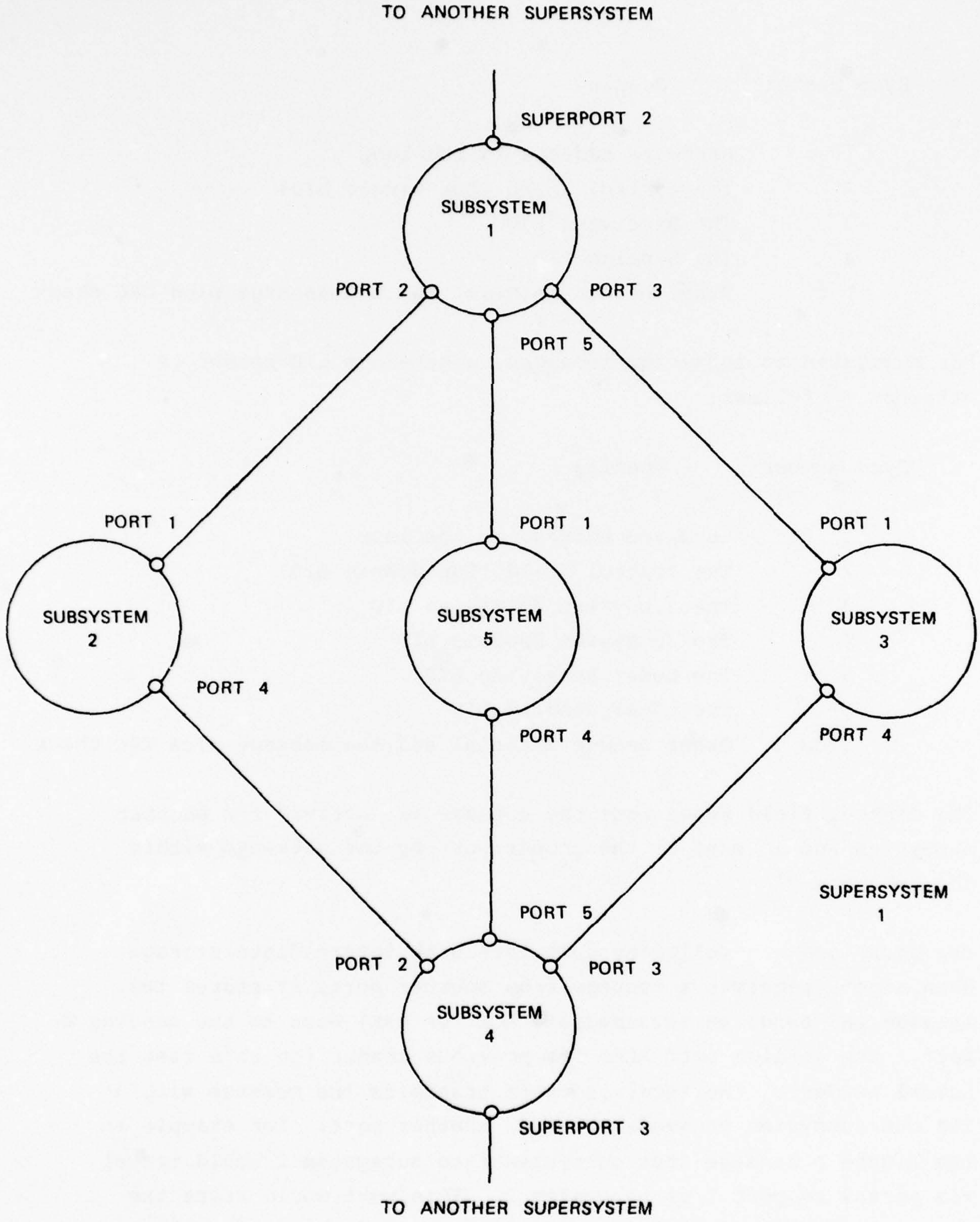


Figure 1.1-1

Supersystem of Connected Subsystens

Byte Number	Meaning
1	Hardware Address in the loop
2	The control field (for single LID)
3	The Receiving LID
4	The Sending LID
5 on	Other header material and the message plus CRC check

For subsystem to subsystem messages, a two-deep LID header is attached as follows:

Byte Number	Meaning
1	Hardware Address in the loop
2	The control field (for double LID)
3	The Subsystem Receiving LID
4	The Subsystem Sending LID
5	The Local Receiving LID
6	The Local Sending LID
7 on	Other header material and the message plus CRC check

The control field shows that the message is destined for another subsystem and is sent to the proper port by the gateways within the subsystem.

The ports are controlled by computers with intermediate storage. When a port receives a message from another port, it stores the message and sends an intermediate ACK (or NAK) back to the sending port. The sending port ACKs its previous sender (in this case the actual sender). The receiving port transmits the message within its own subsystem or sends it on to another port. For example in the figure a message from subsystem 1 to subsystem 2 would travel via port 2 to port 1 of subsystem 2. This port would store the message and ACK the sending port which would transmit the ACK to the sending station. The message would then pass into subsystem 2 to the ultimate receiver which would ACK port 1 to erase the message from in-transit storage with journaling if desired. If the receiver was a host, it would perform a host-to-host ACK by reversing the LID's in pairs.

A message from subsystem 1 to subsystem 4 has three possible paths. Suppose the path via subsystem 5 were the path used.

The message would proceed from subsystem 1 to subsystem 5 with port 1 performing the store, ACK to the sender and transmit to port 4 functions. Port 5 of subsystem 4 would perform the store, ACK to port 1 and transmit to receiver functions. The receiver node would ACK to port 5 and hand the message to the receiving host or terminal. A host-to-host ACK could then be performed.

When more than 250 subsystems are required, partitioning into supersystems is indicatd with superports to other supersystems. Messages between supersystems would require triple pairs of LID's, but the procedure for transmission would be essentially the same as for subsystems.

ATTACH commands would be of the form ATTACH xxx-yyy-zzz where xxx is the local LID in the receiving subsystem, yyy (if used) is the subsystem LID and zzz (if used) is the supersystem address.

The table below shows the largest number of unique LIDs that can be contained in a subsystem, a supersystem and a system of supersystems based on 250 usable LID locations per echelon.

Maximum LIDs	Type of System
250	Subsystem
62,500	Supersystem
15,625,000	System of Supersystems

1.1.3 Application to System Control

If a System Control Architecture similar to ATEC is used, then each station and its node form a subsystem. Since each station might have ten to fifteen LID's, only about fifteen stations can report to a node if a limit of 250 is imposed. For this reason, a larger (12 bit) address might be used to permit up to 4000 LID's per subsystem.

A group of nodes and their sector would form a supersystem with ports connecting them together. Similarly, the sectors and their ACOC would be connected via superports which would form a geographical system. ATTACH and DETACH commands could be used to provide connection from any terminal to any other terminal or any host within the system. Suitable protection mechanisms are assumed (see the security task). In addition to the ATTACH and DETACH commands, it is assumed that the data management language would contain data-set transfer commands as well as those indicated in section 1.2.

1.2 Distributed Data Base Design

The following discussion covers task 1.4 "Design of Generalized Distributed Data Base Algorithms" and task 1.2 "User Language Augmentation Design". Introductory material is provided in the Specification volume of the ESM final report dated April 1977. This specification provides background on computer to computer message formats for distributed file mainpulation. ESM material concerns distributed files; the information below concerns distributed data bases.

TOTAL/PDP-11 of Cincom Systems, Inc. is used as the single computer data base system and the TOTAL/PDP-11, Release 1.1 handbook forms a part of this discussion. Its parlance is used where applicable. The symbol "TDBS" will hereinafter be used in place of "TOTAL/PDP-11".

The TDBS is designed for a single computer with user commands given at the computer terminal. The system described below presents the manner in which a data base may be distributed over a number of computers using a TDBS on each. Furthermore, the user commands are to be applied to any one of a distributed set of ESM terminals connected to the network rather than terminals connected to individual computers. Commands applied to these distributed terminals provide access to the entire distributed data base in a user-transparent fashion. The distributed data base management system will be referred to as the DDMS.

The DDMS provides not only a distributed data base but permits duplication of records in a fashion to ensure that each record in the system is duplicated in a retrievable manner and that some of the duplication can be supplied by computers that are not necessarily equipped with a TDBS.

The explanation of how these ends are accomplished is given in terms of the specific example of circuits and trunks and their cross-referencing.

1.2.1 Single Computer Data Base System

The system control data base system used for the example is made up of circuits and trunks. In this discussion a circuit is defined as the logical entity that connects two terminals. A trunk is the physical entity that connects two stations. A segment is the logical entity connecting two stations. A trunk is divided into channels of smaller bandwidth. Thus a circuit as it spans many segments may be made up of a group of trunks, and a trunk at a given segment may be made up of a group of circuits assigned to various channels.

The manner in which a data-base consisting of circuit descriptions, trunk descriptions and the two cross-references of circuits contained in a trunk at a given segment and the trunks that make up a circuit over the full run of each circuit involving many segments will be shown in TDBS terms. This example is chosen because of the two cross-references required. Although the example is described first for a single computer, it is arranged in a fashion to provide for easy expansion to a multiple computer system.

The data-base is described in terms of five data-sets. Two are master-data-sets and three are variable-data-sets. There is a master-data-set for circuits and one for trunks. Each consists of short records that show only keys and links and will be called the "key-sets". In general, there will be a key-set for each type of entity in the system. Two of the three variable-data-sets are the "description-sets", one for circuits and one for trunks. These are typically sets with long records that describe the entities involved in as much detail as required. In general, there will be a description-set for each type of entity in the system. The third variable-data-set is the "linkage-set" that provides for two fold cross-referencing. Hierarchical cross-referencing links directly from the higher master-data-set to the lower variable-data-set and will not be shown.

Linkage-sets will generally supply the twofold cross-references per set. Linkage-sets are composed of short records that supply only the cross-reference pointers and sometimes a small amount of data associated intimately with the cross-reference itself.

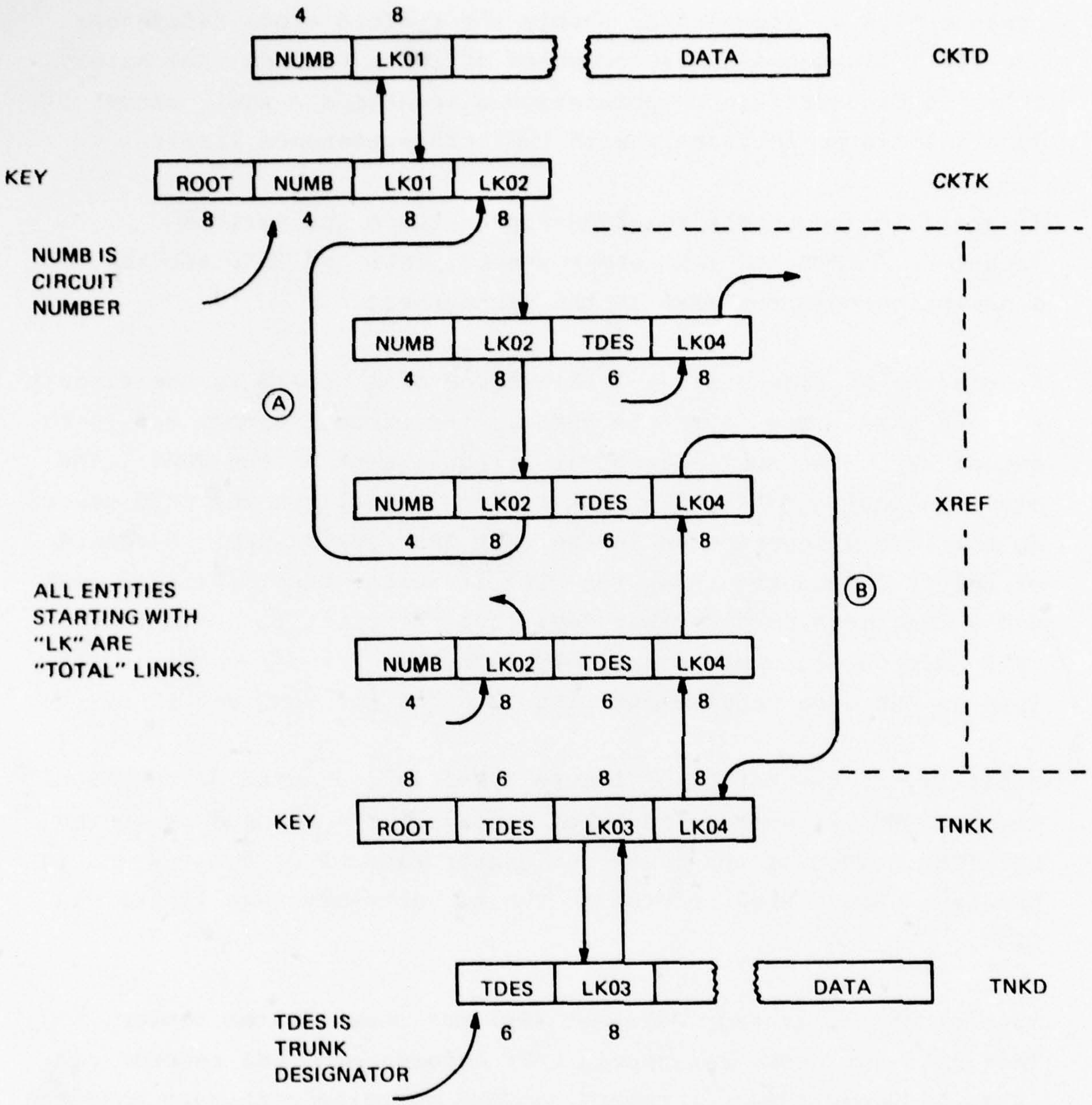
Figure 1.2-1 shows the relationships between the various data-sets. CKTK and TNKK are key-sets, CKTD and TNKD are the description-sets and XREF is the linkage-set.

At the top of figure 1.2-1 is a diagram of a record in the circuit key-set CKTK. The four byte NUMB is the circuit number and is the actual key. The ROOT element is a requirement of the TDBS. The elements LK01 and LK02 are TDBS links. LK01 links the CKTK record to the record description in the CKTD description-set. A record of the CKTD data-set shows the circuit number NUMB, the link LK01 and a long data portion that describes the circuit. Note that CKTK records are short using only 28 bytes. CKTD records are as long as the data requirement plus 12 bytes for NUMB and LK01.

Similarly at the bottom of figure 1.2-1 is a diagram of the trunk key-set TNKK with its associated record of the trunk description-set TNKD. TDES is the trunk designator element of 6 bytes and is the key. Associated records in the two data-sets are linked via LK03.

Records of the linkage data-set XREF are shown in the center. This data-set cross-references CKTK records and TNKK records via links LK02 and LK04. A record in XREF contains both keys NUMB and TDES as well as the links LK02 AND LK04. A single XREF record associates a circuit and a trunk in two ways. It shows the trunk as part of the chain of trunks that comprise a circuit and it shows the circuit as a member of the group of circuits that populate the channels of a trunk.

If one starts at a CKTK record, a series of READV commands (see the TDBS manual) on link LK02 will follow the path marked A in the figure and will provide the TDES keys for all of the chain of trunks in the circuit. If one starts at a TNKK record, a series



LINKAGE (A) CONNECTS A CIRCUIT KEY TO ALL TRUNKS THAT MAKE UP THE CIRCUIT FROM END-TO-END.

LINKAGE (B) CONNECTS A TRUNK KEY TO ALL CIRCUITS CONTAINED *WITHIN THE TRUNK'.*

Figure 1.2-1
Diagram of Circuit-Trunk Data-Set

of READV commands on LK04 will follow the path marked B in the figure and will provide the NUMB for each circuit that populates the channels of that trunk. If the XREF records are supplied with the additional data giving the channel used, then following path B will also supply the trunk channels occupied by circuits. Thus, each XREF record is 30 bytes plus channel data.

1.2.2 Data Definition Example

The data-base described above is defined below using the Data Base Definition Language of the TDBS. The data portions are arbitrarily set to a size of 70 bytes. The actual size depends on the amount of data required.

```
BEGIN-DATA-BASE-GENERATION
DATA-BASE-NAME=CKTTNK
OPTIONS:LOG=N, OUTPUT=D
SHARE-IO
IOAREA=MAS1
IOAREA=MAS2
IOAREA=VAR1
IOAREA=VAR2
IOAREA=VAR3
END-IO

BEGIN-MASTER-DATA-SET
DATA-SET-NAME=CKTK
IOAREA=MAS1
CKTKROOT=8
CKTKCTRL=4               CIRCUIT NUMBER IS KEY
CKTKLK01=8               LINK TO CIRCUIT DESCRIPTION
CKTKLK02=8               LINK TO XREF DATA-SET
END-DATA
UIC=[1,4]
DEVICE=RK05
LOGICAL-RECORD-LENGTH=28
LOGICAL-RECORD-PER-BLOCK=73
TOTAL-LOGICAL-RECORDS=146
DRIVE=12, 8, SYO
SEND-MASTER-DATA-SET
```

```
BEGIN-MASTER-DATA-SET
DATA-SET-NAME=TNKK
IOAREA=MAS2
TNKKROOT=8
TNKKCTRL=6                 TRUNK DESIGNATOR IS KEY
TNKKLK03=8                 LINK TO TRUNK DESCRIPTION
TNKKLK04=8                 LINK TO XREF DATA-SET
END-DATA
UIC=[1,4]
DEVICE=RK05
LOGICAL-RECORD-LENGTH=30
LGOICAL-RECORDS-PER-BLOCK=34
TOTAL-LOGICAL-RECORDS=136
DRIVE=13,8,SYO
END-MASTER-DATA-SET

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=CKTD
IOAREA=VAR1
BASE-DATA:
CKTDCODE=2                REQUIRED BY LANGUAGE
CKTDNUM=4                THE CIRCUIT NUMBER
CKTKLK01=8=CKTDNUMB       LINK TO CIRCUIT MASTER
CKTDDATA=76               THE DATA ASSOCIATED WITH THE CIRCUIT
END-DATA
UIC=[1,4]
DEVICE=RKO5
LOGICAL-RECORD-LENGTH=90
LOGICAL-RECORDS-PER-BLOCK=17
TOTAL-LOGICAL-RECORDS=153
DRIVE=14,27,SYO
END-VARIABLE-ENTRY-DATA-SET
```

```
BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=TNKD
IOAREA=VAR2
BASE-DATA:
TNKDCODE=2
TNKDTDES=6                   TRUNK DESIGNATOR
TNKKLK03=8=TNKDTDES          LINK TO TRUNK MASTER
TNKDDATA=74                  THE DATA ASSOCIATED WITH THE TRUNK
END-DATA
UIC=[1,4]
DEVICE=RK05
LOGICAL-RECORD-LENGTH=90
LOGICAL-RECORDS-PER-BLOCK=17
LOGICAL-RECORD-RECORDS=136
DRIVE=15,24,SYO
END-VARIABLE-ENTRY-DATA-SET

BEGIN-VARIABLE-ENTRY-DATA-SET
DATA-SET-NAME=XREF
IOAREA=VAR3
BASE-DATA
XREFCODE=2
XREFNUMB=4                   CIRCUIT NUMBER
XREFTDES=6                   TRUNK NUMBER
CKTKLK02=8=XREFNUMB          THE LINK-PATH A
CKTKLK04=8=XREFTDES          THE LINK-PATH B
XREFCHAN=4                   OPTIONAL CHANNEL NUMBER
END-DATA
UIC=[1,4]
DEVICE=RK05
LOGICAL-RECORD-LENGTH=32
LOGICAL-RECORDS-PER-BLOCK=16
TOTAL-LOGICAL-RECORDS=720
DRIVE=16,45,SYO
END-VARIABLE-ENTRY-DATA-SET
END-DATA-BASE-GENERATION
```

1.2.3 Double Computer DDMS with Duplication

For distributing the data-base between two computers, the data-base definitions on the two are indentical and are the same as that of the single computer system The difference is that some of the records are distributed between the two. The key data-set records are fully duplicated on the two computers. The two variable data-sets CKTD and TNKD have their records shared between the two computers. The XREF records are fully duplicated. Note that the short records are duplicated and the long records are shared.

Suppose that such a distributed data base exists and that a user enters a circuit key request in the form of a READM on CKTK at his terminal. The request goes to his dialog director via the network and the key is accessed via the TDBS on the director. The key, if present in the system, is present at the director because all keys exist on both computers. The user gets a "key present" return. He then enters a READV for that key using link CKTKLK01. If the corresponding record exists in the CKTD data set on the director, the data is sent to the user. If not, a host-to-host message (see the ESM Specification) is sent to the other computer which accesses the record and sends it to the user.

The user now sends a group of READV commands on CKTKLK02. Since XREF records are fully resident on both computers, the user receives the full set of TDES keys associated with that circuit. If he desires information on any of the trunks, he issues a READM command on TNKK for the TDES key involved. The trunk must exist so he follows with a READV on TNKKLK03 for the trunk description. This may involve a host-to-host message. He then may issue a series of READV on TNKKLK04 commands to obtain the keys and channels for the circuits that go through that trunk.

If one of the computers is out of service all of the keys and all XREF records are available on one computer. Only a part of the descriptions are available on one computer. Since the descriptions are single key records, they can be duplicated on a third computer that does not support the TDBS. They may be stored in the form of regular files that are accessed by means of hash-codes or other standard methods. In this way, the data base is fully duplicated and available as long as any two out of three computers is in operation. An example of this could be embodied in the ESMD which has two PDP-11/40 computers with the TDBS installed and one Burroughs B776 without the TDBS. Such a system could be termed a three computer fully duplicated DDMS.

1.2.4 User Language Augmentation

Before extending the data-base discussion to include data-base initiation and distribution among larger numbers of computers, augmentation to the user language for data base commands are discussed below. This discussion covers task 1.2.

The augmentation consists of commands in the form of a single string of characters to be entered at any of the three loop-connected terminals of the ESM. Terminals directly connected to computers cannot be used. When mode 4 of the user language is selected, the CRT involved will show one or two read-protected regions. One such will always exist on the top line, the other (when used) will be for the purpose of receiving data on read commands, revising previously read data on write commands or entering new data on add commands. The first region is the command region and the second region is the data region. When data is presented in the data region it will be shown as one line underneath a computer-supplied header display.

1.2.4.1 Master Data-Set Commands

Master data-set commands are given in the form of a string 10+K bytes long starting at the left forms mark of the command area and presented without intervening spaces. The value K is the length of the key for the data-set involved. The first six bytes are the commands TREADM, TWRITM, TADD-M or TDEL-M. The next four bytes are the name of the data-set (always four bytes). The key follows the data-set name. Thus, the string TREADMCKTK6637 is the read command for master data-set CKTK with a circuit number of 6637.

1.2.4.1.1 The TREADM Command

The read master data-set command TREADM finds the record (if present) and presents the data in the data region. The command remains in the command field. There are three exceptions as follows:

BCTL	The control key contains spaces
MRNF	The record does not exist
INVD	The command contains an error

1.2.4.1.2 The TWRITM Command

This command writes the revised data region of a record obtained from a TREADM command. To do this, perform the TREADM, change the data region as required and change the TREADM to TWRITM. Be sure to place the CRT cursor at the first character of the command region by pressing the HOME key before the XMT key is pressed. Exception codes are:

BCTL, INVD as in TREAD
UCTL The control key in the command and the control file in the data do not match.

Note that in the two master data-sets of the example showing CKTK and TNKK data-sets, the only data in the record is the key itself. The links do not form part of the data. In a more general situation, however, other data may be included.

1.2.4.1.3 The TADD-M Command

This command adds a record to a master data-set. Perform a TREADM command first to be sure that the key does not already exist. This will return a MRNF exception code and display an empty data region with headings. The TADD-M command and data may then be entered. Exception codes are as follows:

BCTL, INVD, UCTL as in TREADM and TWRITM
DUPM The record key already exists
FULL There is no more space available

1.2.4.1.4 The TDEL-M Command

This command deletes an existing record. It is a good idea to read the record first through a TREADM command to be sure that the record involved is the correct one. Note that if any variable data-sets are linked to the master record, no deletion can occur. Exception codes are as follows:

BCTL, INVD, MRNF as in TREADM
IMDL Variable records are linked to this record.

1.2.4.2 Variable-Entry Data-Set Commands

The Variable-entry data-set commands are given in the form of a string 18+K long starting at the left forms mark of the command area and presented without intervening spaces. The value K is the length of the master key to which the data-set is linked. The first six bytes are the comands TREADV, TREADR, TWRITV, TDELVD, TADDVA and TADDVC. The next four bytes are the name of the data-set. The next eight bytes are the linkage-path in the form NAMELKxx where NAME is the name of the master data-set for that linkage-path. This is followed by the value of the key of the master data-set involved. The key length is K. Thus, for a description of the circuit called by a TREADMCKTKA234 command, the call TREADVCKTDCKTKLK01A234 is given. TREADV is the command, CKTD is the set name, CKTKLK01 is the link-path, and A234 is the key in CKTK to which the variable-entry data-set CKTD is linked. For an explanation of such linkage paths, refer to the TOTAL manual.

1.2.4.2.1 The TREADV Command

This command follows link-path pointers starting from the first variable record pointed to by the link specified in the master record specified. Repeated TREADV will point to succeeding variable records in the chain of the link-path until an "END." shows in the exception space to indicate that the chain has ended. For code-directed reads, add *CODE=xx to the eighteen byte command string to form a twenty-six byte command string. For code-directed reads, see Appendix A of the TOTAL manual. Code-direction is not used in the example given in 1.2.2 above. Exception codes are as follows:

BCTL:	A control field contains blanks
MLNF:	The linkage-path name is not valid
IVRC:	The code-direction contains an undefined value.
MRNF:	The related master-record cannot be found.
INVD:	The command contains some other error.

1.2.4.2.2 The TREADR Command

This command is the same as TREADV except it proceeds along the chain backwards. In general, do not execute TDELVD commands after TREADR. Use TDELVD only after TREADV. Exception codes are the same as TREADV.

1.2.4.2.3 The TWRITV Command

This command follows a TREADV or TREADR command to change the data in the data region. No controls may be changed. Only the data may be changed and TWRITV substituted for TREADV or TREADR. Any change in linkage-path may cause serious problems in linkage-path maintenance.

The exception code is INVD only.

1.2.4.2.4 The TDELVD Command

After a TREADV, change the command to TDELVD to delete the record. The complete record will be deleted and all linkage-paths will be changed accordingly. Another TREADV after the TDELVD will provide the next record in the chain. If a TREADR follows the TDELVD, the next preceding record in the chain of the deleted record will be skipped and the preceding record before that will be read. Exception codes are as follows: IVRC, MLNF and INVD as in the TREADV command.

1.2.4.2.5 The TADDVA Command

This command should be given after a TREADV command. It adds a variable record to the chain immediately after the last TREADV or at the start of the chain after a TREADM command. This is done only on the chain of the link specified. On all other links, the addition is at the ends of the respective chains.

The exception codes are as follows:

BCTL,	IVRC, MLNF and INVD are the same as for TREADV.
FULL	The data-set has no room for this record on any computer that holds the data-set.
MRNF	A master record does not exist for a control field (primary-link)
UCTL	A control-key field does not match the corresponding field in the data area
NSMR	A master record does not exist for a control field (secondary-link).

1.2.4.2.6 The TADDVC Command

This command is similar to TADDVA except that the record is added at the end of all linkage path chains for the record. Exception codes are the same as TADDVA.

1.2.4.3 The TRDNXT Command

This function is used to determine the records that exist in a file in the form of a serial retrieval. A series of TRDNXT commands will retrieve records until the end of the data-set is reached. An "END." display will then be shown. The command has the form TRDNXTNAME where NAME is the data-set name.

1.2.5 Establishing a Data-Base

Specific directions for establishing a TDBS data base structure are given in the TOTAL handbook and the ESMD user manual. In essence, however, the steps are as follows:

(a) Set up a data-base-generation file as shown in 1.2.2 above using the PDP11 Editor Utility EDI. Call this DBMODB.DBG.

(b) Run the task DBGEN using this file to form an assembly code file DBMODB.MAC and assemble the DBMODB.MAC into the object file DBMODB.OBJ.

(c) Establish a command file that includes the object files DBMODB.OBJ, TOTAL.OBJ, DATBAS.OBJ (TOTAL and DATBAS are part of the TOTAL software) and the object files that make up the user language. This command file (call it USRLNX.CMD) defines the task building parameters.

(d) Build the user language task USRLNG.TSK by means of TKB and the command file USRLNX.CMD.

(e) Run the task DBFMT with DBMODB.OBJ and file names that have the same names as the data-set names. This sets up the actual files that will store the data in the required form.

The result is a task USRLNG that can be run in a fashion to operate on these files through the TOTAL module in a fashion consistent with the data-base structure.

The operation described should be performed on both PDP-11/40 hosts.

1.2.5.1 Entering the Data

In the example in section 1.2.2 and figure 1.2-1 the data-sets TNKK and TNKD represent the physical system of trunks in a geographical area. To enter a new trunk to the system, perform a command group as follows:

(a) TREADMTNKK newkey where newkey is the trunk designator. Since the record does not exist a MRNF code is returned and an empty data region with headings appears.

(b) TADD-MTNKK newkey with the newkey value in the data region writes the new master record.

(c) TPEADVTNKDTNKKLK03 newkey will result in an "END." return with a data region defined for the description of the trunk.

(d) TADDVCTNKDTNKKLK03 newkey and the data describing the trunk in the data region writes the new variable record. Any subsequent TREADM followed by TREADV on that key will produce the data on the trunk in the data region.

Perform as many command groups as desired to establish the trunk data-set for the geographical area. Each record of TNKK will reside on both PDP-11/40 hosts. Each record of TNKD will reside on the PDP-11/40 that is the dialog director with a copy on the B776. If host A is the dialog director and a distribution of TNKD data is desired, then an ATTACH B command should be issued to make host B the director for part of the record entry. If the B776 (host C) is the director or if the LSI-11 program development unit (host D) of loop 4 is the director, then perform an ATTACH A or ATTACH B command.

To enter circuit data, perform the same group of commands as for trunks for each circuit except use the CKTK and CKTD data-sets with link CKTKLK01. In addition, set up the linkage-set XREF for each circuit as follows:

(a) Perform a TREADMCKTKkey for the circuit key.

(b) Perform a TREADVXREFCKTKLK02ckey which will cause an "END." to be returned with a data region of the XREF data-set.

(c) Perform a set of TADDVCXREFCKTKLK02ckey commands with the circuit key and a trunk key in the data area. The circuit key must be ckey.

The trunk key will change with each command until all trunks that form the circuit have been entered. The links TNKKLK04 for the trunks involved will automatically be linked to the TNKK record. If desired, the trunk channel can also be named in the XREF data area. This might be entered later as shown below.

The XREF records exist on both hosts A and B.

All of the record duplication required is handled automatically by the user language through host-to-host messages.

Once the XREF data-set is established, the channels may be set for each trunk as follows:

(a) Perform a TREADMTNKKtnkkey.

(b) Perform a TREADVXREFTNKKLK04tnkkey to get the data on the first XREF record on the LK04 path.

(c) Perform a TWRITVXREFTNKKLK04tnkkey with the channel number added to the data area.

(d) Repeat (b) and (c) until an "END." return is obtained.

1.2.6 Host-to-Host Messages

Host-to-Host Messages involving data-base have the form shown below. For background on host-to-host messages see the specification volume of the ESM final report.

Word Index	Meaning
1	Dl-D2 of header
2	D3-D4 of header
3	Logical ID's of sender and receiver hosts
4	The access command hexadecimal 001E or 001F
5	Either hexadecimal 0000 or the logical ID of the receiving CRT in binary form
6 on	The message text followed by end-of-packet.

The access command 001E hex (30 decimal) is for the data-base host-to-host (DBHH) message and the access'command 001F hex (31 decimal) is for host-to-host ACKs or NAKs (ANHH). Every DBHH is answered by an ANHH in the form

Word Index	Meaning
1	Dl-D2 of the header
2	D3-D4 of the header
3	Logical ID's reversed
4	001F hex
5	0000 hex
6-7	ACK or NAK (ASCII)
8	End-of-packet.

When all hosts are operating, DBHH messages are sent only by the dialog director for TWRITM, TADD-M and TDEL-M for master records and TWRITV, TDELVD, TADDVA and TADDVC for variable records. For some variable records (that are not present in the director), TREADV and TREADR commands are sent as well. The write, add and delete commands are always sent to ensure that the records are duplicated. TRDNXT commands are never sent as DBHH messages.

The format used is that shown above with the access code 001E hex (30 decimal). The CRT logical ID is always included so that data and exception messages may be sent to the CRT by the receiving host. The message text is identical to that sent by the CRT.

The receiving host always sends an ACK or NAK back to the sending host. A NAK causes the message to be retransmitted by the sending host. Lack of either after a suitable timeout results in an exception condition.

1.2.6.1 Exception Conditions in DBHH Messages

A lack of an ANHH causes at least one retransmission by the sending host. After a prescribed number of retransmissions, the sending host sends a "host not reachable" message to the CRT with the logical ID of the receiving host. Alternate path attempts to reach the receiving host will have been tried by the CIE or CIP. The sending host should also send "host not reachable" messages to the other hosts in the system. The host not reachable status is stored in the sending computer. The sending computer thereafter performs in exception mode for the host involved. The exception modes are described below for DBHH messages.

For read types of DBHH messages, the back-up host is the recipient. For write, delete and add type messages, the messages are stored on "mailbox" disk until a message is received that established the host as reachable. The messages are then sent.

1.2.7 Use of Several Computers

In order to distribute a data-base among several computers, all of the master data-sets for keys must exist on all of the computers that hold the data-base. On computers that do not hold the data-base, a file may be included to provide information for the relaying of requests to computers that do. Where a reasonably large number of computers exist, it becomes inefficient to do a single data base over all computers. If multiple data-bases exist, then each data-base may be assigned to a small group of computers.

The variable-entry-data-sets may be distributed among the computers that hold the data-base duplicate records distributed in a way to ensure that every record exists on at least two computers. Duplicates of single-key descriptive variable records may be stored on machines that are not involved with the data-base. Multiple-key variable records must be stored on data-base computers.

An example might be the use of four data-base computers A,B,C,D and two non-data-base computers E,F storing the data-base described previously. The full CKTK and TNKK key master-data-sets would reside on A,B,C,D. The CKTD and TNKD description files would be distributed among the four with records on A and B copied on E and records on C and D copied on machine F. The linkage data-set XREF would be distributed among A,B,C,D. There would also be a file XREC that would be identical with XREF except that XREF records on A would be copied in XREC on B, XREF records on B would be copied in XREC on C, etc.

The data-base would then be fully duplicated but distributed among a number of machines thereby providing a highly survivable data-base structure.

2. Modeling Fail-Soft Architectures

2.1 System Reliability

2.1.1 System Control Reliability Modeling

The ESM multiloop system can be used to model and investigate various System Control architectures with respect to reliability. Various System Control configurations can be modeled where loops can represent sites, ESM processors can perform System Control functions, and ESM links can represent DCS orderwires. The ESM loops can represent various levels of the DCS System Control hierarchy (i.e., DCAOC, ACOC, Sector, Node and Station). The ESM network architecture featuring addressing by process name implemented by the nodal Logical ID/Functional Address (LID/FAD) conversion tables allows logical definition of many different network configurations

By defining the nodal LID/FAD tables such that the network configuration simulates various System Control architectures, experiments can be performed to investigate the reliability of these systems. In the original ESM three-loop network, available resources include two host processors and three terminals. The addition of the ESMD loop 4 supplies an additional host processor and terminal, and eight microcomputers each with 32K bytes of memory that are programmable in a higher-level language (extended ALGOL). These nodal microcomputers can represent nodes in a simulated System Control network configuration. The MSCDM loop 5 will provide eight microcomputers programmable in FORTRAN each with 64K bytes of memory. One of the microcomputers will have mini-disk enabling it to act as a host processor; two terminals will be included with the system. Thus the five loop system will allow System Control network simulations of up to 18 processors. An additional 11 processors are available with the original 3 loop ESM nodal microprocessors, however the lack of a higher-level language and limited control memory results in a more difficult implementation.

The original three loop ESM configuration is given in Figure 2.1-1a. A computer network can be modeled by a linear graph G=(N,E) where N is a nonempty set of n nodes that corresponds to the computer centers in the network and E is a set of b edges or branches that corresponds to the communication links. If we represent the ESM loops as nodes, then the graph model for the three loop ESM configuration is given in Figure 2.1-1b. Any graph is said to be connected if there is at least one path between every pair of nodes n_i, $n_i \in N$. The three loop ESM configuration is fully connected since each node is connected to every other node over a single link. The network also possesses the characteristic of two-connectivity since each node has at least two links connected to it.

Reliability is the characteristic of an element of a physical system, expressed by the probability that it will perform a required function within established limits of tolerance, under the operating conditions encountered for a stated period of time. Failure is the event after whose occurrence the system violates the permissible limits of performance (2-1).

The study of the possible failure of network elements and the subsequent overall degradation of network performance is called the survivability or vulnerability problem. Network survivability can be divided into two areas: deterministic survivability and probabilistic survivability. In the deterministic case, a complete knowledge of the system to be tested is assumed and a deterministic damage strategy is used; in the probabilistic case, a probabilistic graph is associated with the physical system such that probabilities that its elements are operative are assigned to either branches or nodes (2-1).

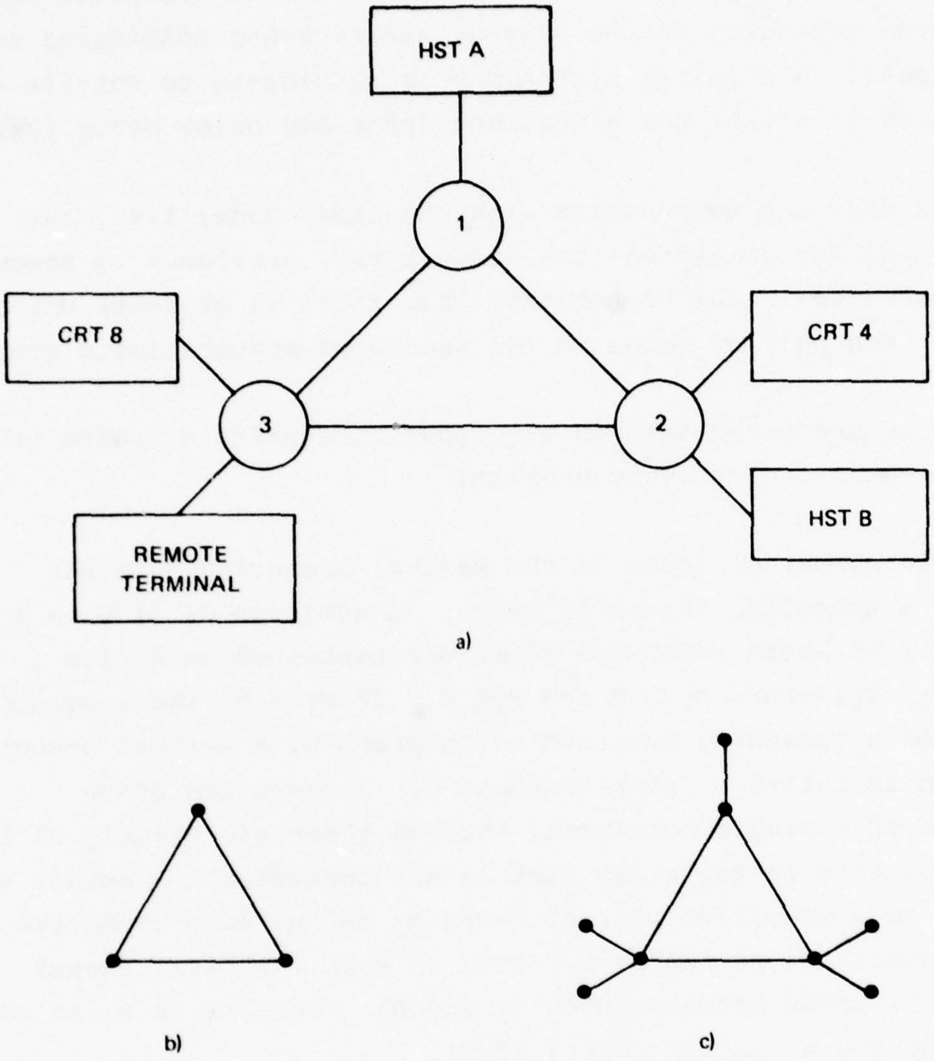


Figure 2.1-1. ESM Three Loop Network

To study the vulnerability of a system a failure criterion must be formulated depending on the type of system being considered and its purpose. A physical system may be considered to survive an experiment if any of the situations described below occur (2-1):

i) Each node can communicate with any other node; i.e., the network remains connected (the connectivity problem.) A measure of performance is the probability that there is at least one path between each pair of nodes in the asociated probabilistic graph.

ii) There are paths between some specified pairs of nodes (the terminal-pair connectivity problem).

iii) The number of nodes in the maximal connected subgraph exceeds a specified threshold value. A subgraph G_1 of G is a graph all of whose nodes and edges are contained in G, i.e., $G_1 = (N_1, E_1)$ where $N_1 \subseteq N$ and $E_1 \subseteq E$. If $N_1 = N$, the subgraph G_1 is called a "spanning subgraph". In graph G, a maximal connected subgraph is called a "component" of G. A connected graph consists of a single component, whereas there are clearly at least two components in any graph that is not connected. A cutset with respect to a specified pair of nodes n_i and n_j in a connected graph, sometimes called an i-j cut, is such that its removal breaks all paths between nodes n_i and n_j (results in n_i in one component and n_j in the other) (2-2).

iv) The minimum length surviving path between each pair of nodes is no longer than a specified length.

v) The minimum length surviving path between some sp cified pairs of nodes is at most equal to a specified length.

The various components of the ESM system that can fail and the effect of the failure are listed below:

i) ESM Node - An ESM node consists of a Loop Interface Unit (LIU), a Control and Interface Processor (CIP), Memory, and an External Interface Card. The effect of a node failure results in the loss of the device connected to that node. In the original three ESM loops a certain type of node failure can bring down the entire loop. This failure occurs when the LIU acts like an open circuit such that the loop shift register is broken. In the ESMD loop 4, this type of failure does not bring down the loop due to the automatic loop-back feature.

ii) Link - An ESM link connects loops together. A link failure results in a break in communication betwen loops.

iii) Device - The loss of a device disconnects it from the network. Devices in ESM are either processors (e.g., PDP 11/40) or terminals (e.g., TD802 CRT). For ESMD loop 4, the nodes may be viewed as devices when experiments are done in which the CIP's are used as processors rather than intelligent interfaces.

iv) Power Supplies - In the ESM, each loop contains a single power supply. Thus a failure of a power supply results in the loss of the entire loop. In an operational system power supplies would be duplicated in a loop cabinet. In a distributed loop system each node would have its own power supply.

Based on the above component failures a more detailed graph model of the ESM three loop network is given in Figure 2.1-1c. The external devices are represented as graph nodes or vertices which are connected to the loop via a link. These links can be considered to have a high availability as compared to the inter-loop links. The link connected to the remote terminal would have a lower avaiability since the connection is via a leased telephone line.

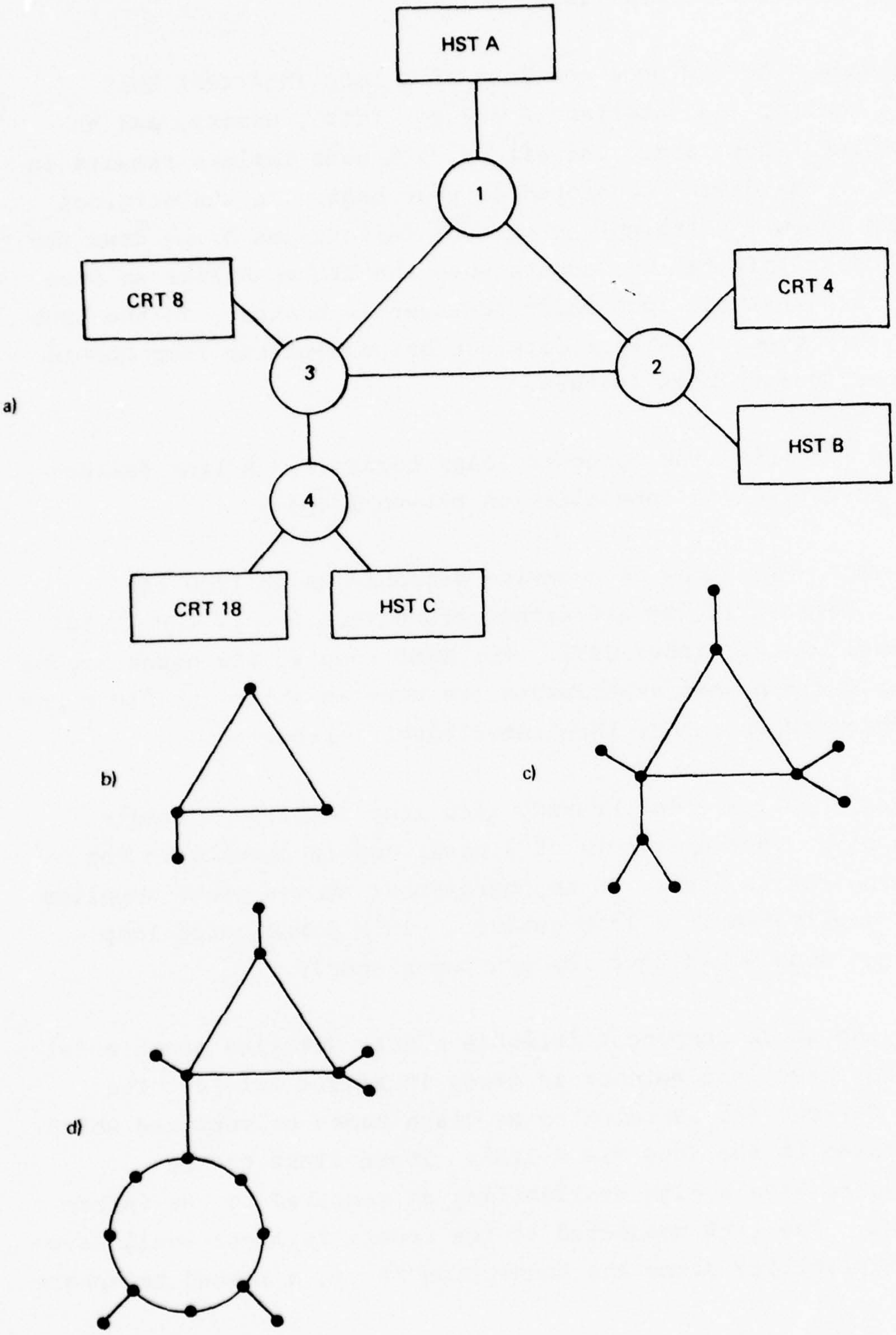


Figure 2.1-2. ESM Four Loop Network

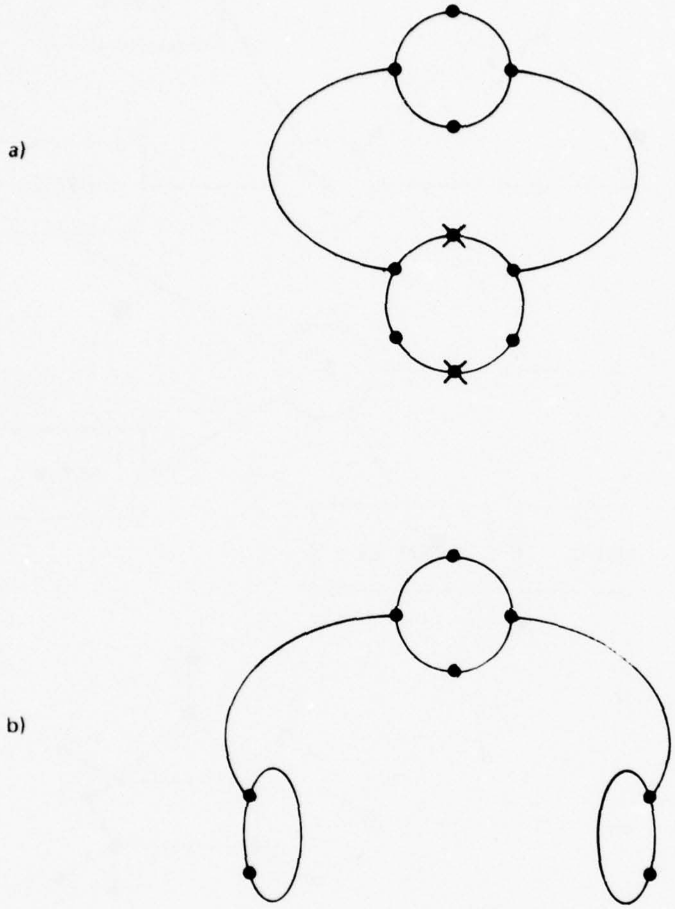


Figure 2.1-3. Two node failure with loop-back and gateway nodes

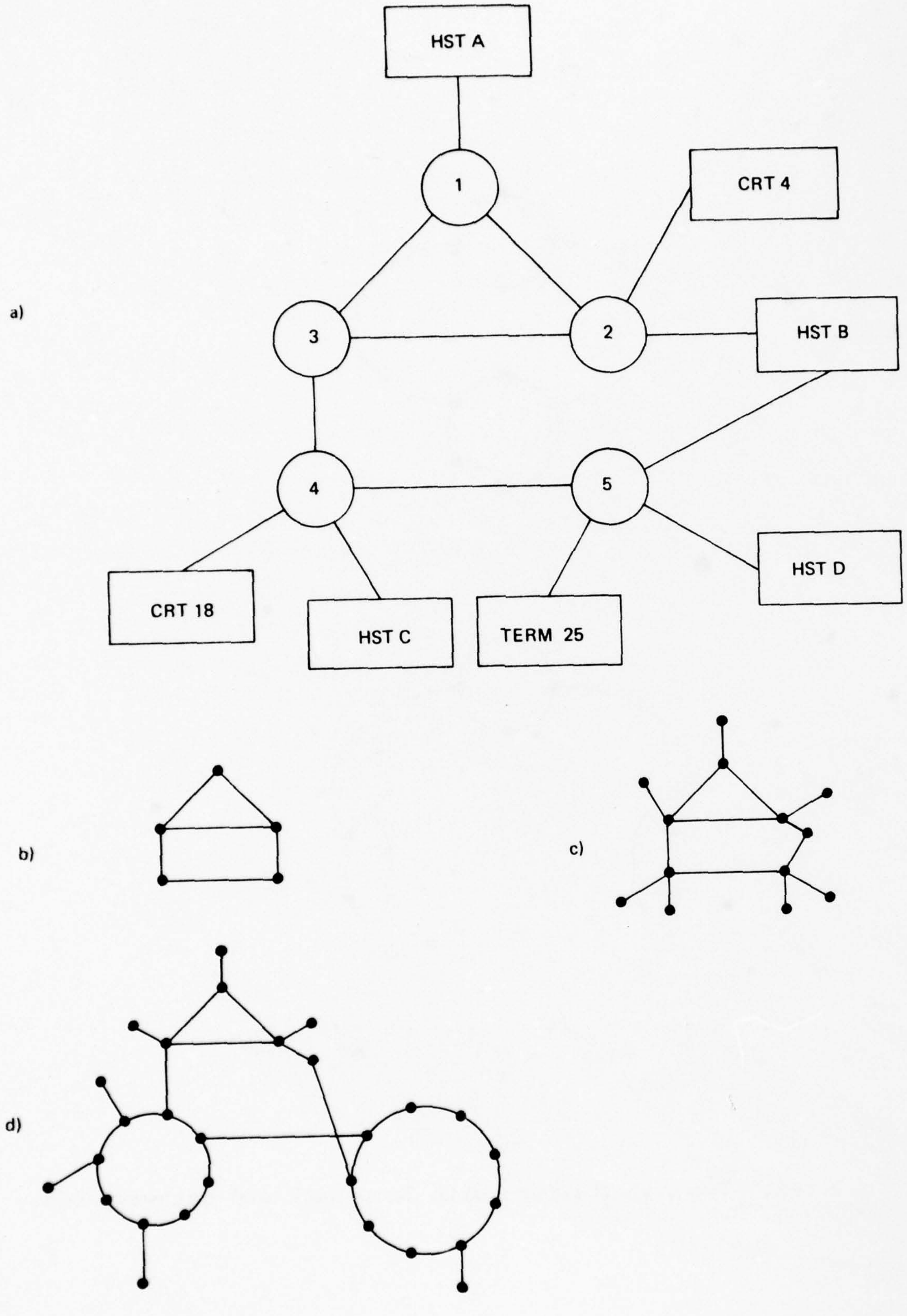


Figure 2.1-4. ESM Five Loop Network

The four loop ESM configuration is given in Figure 2.1-2a. The graph model for the loop interconnections is given in Figure 2.1-2b. Note that the four loop system does not have the property of two-connectivity and is not fully connected. If devices are added to the system, the graph model of Figure 2.1-2c results. If we assume that the nodes of loop 4 are acting as processor devices then the graph model of Figure 2.1-2d results. Note that a single node failure will not bring down loop 4, but a power supply failure will bring down the loop 4 cabinet.

The automatic loop-back feature that is used in loop 4 and described in Section 2.2.4 guarantees that single node failures do not bring down the loop. The system will also tolerate certain multiple node failures. One type of multiple node failure that will not affect the rest of the loop is if adjacent nodes fail. This type of failure could occur as the result of battle damage in a distributed loop where loop-around could occur such that the adjacent set of nodes affected are isolated from the network. Two nodal failures (i.e., failures that result in an open loop) that are not adjacent will result in two independent loops with the failed nodes isolated from the system. If the two independent loops contain gateway nodes, connectivity may still be maintained via the gateway nodes. An example of this situation is given in Figure 2.1-3. Figure 2.1-3a shows the network before the failures and Figure 2.1-3b shows the still connected network with the failed nodes isolated.

Figure 2.1-4 shows the five loop ESM configuration with the LSI-11 MSCDM loop added. Note that the network configuration has two-connectivity. The large amount of nodes and links in this network provides a great deal of flexibility for System Control reliability modeling experiments.

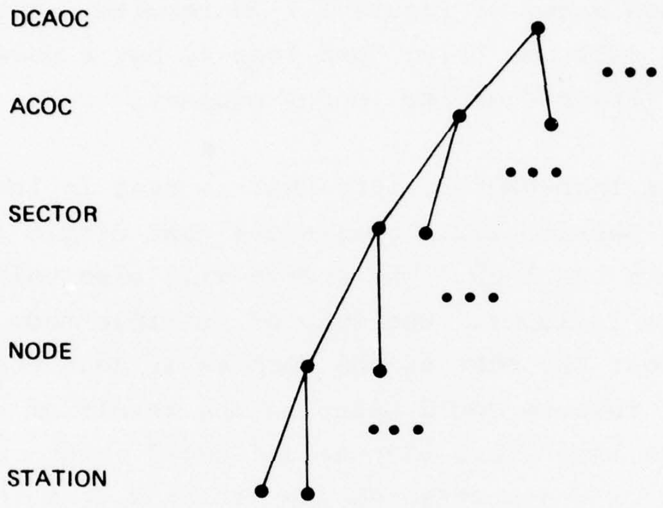


Figure 2.1-5. DCS System Control Hierarchy

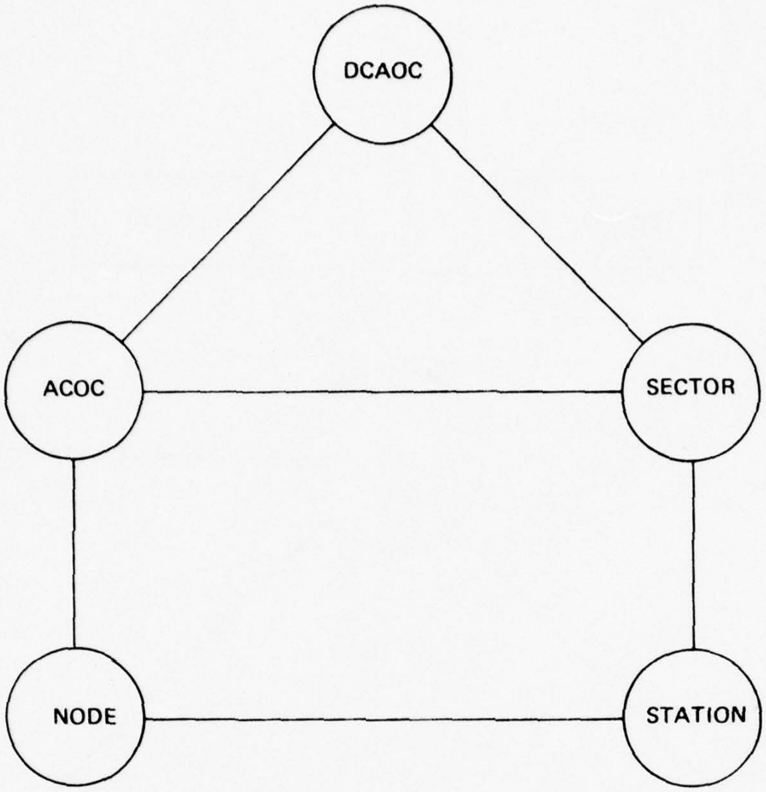


Figure 2.1-6. Individual System Control levels mapped to loops

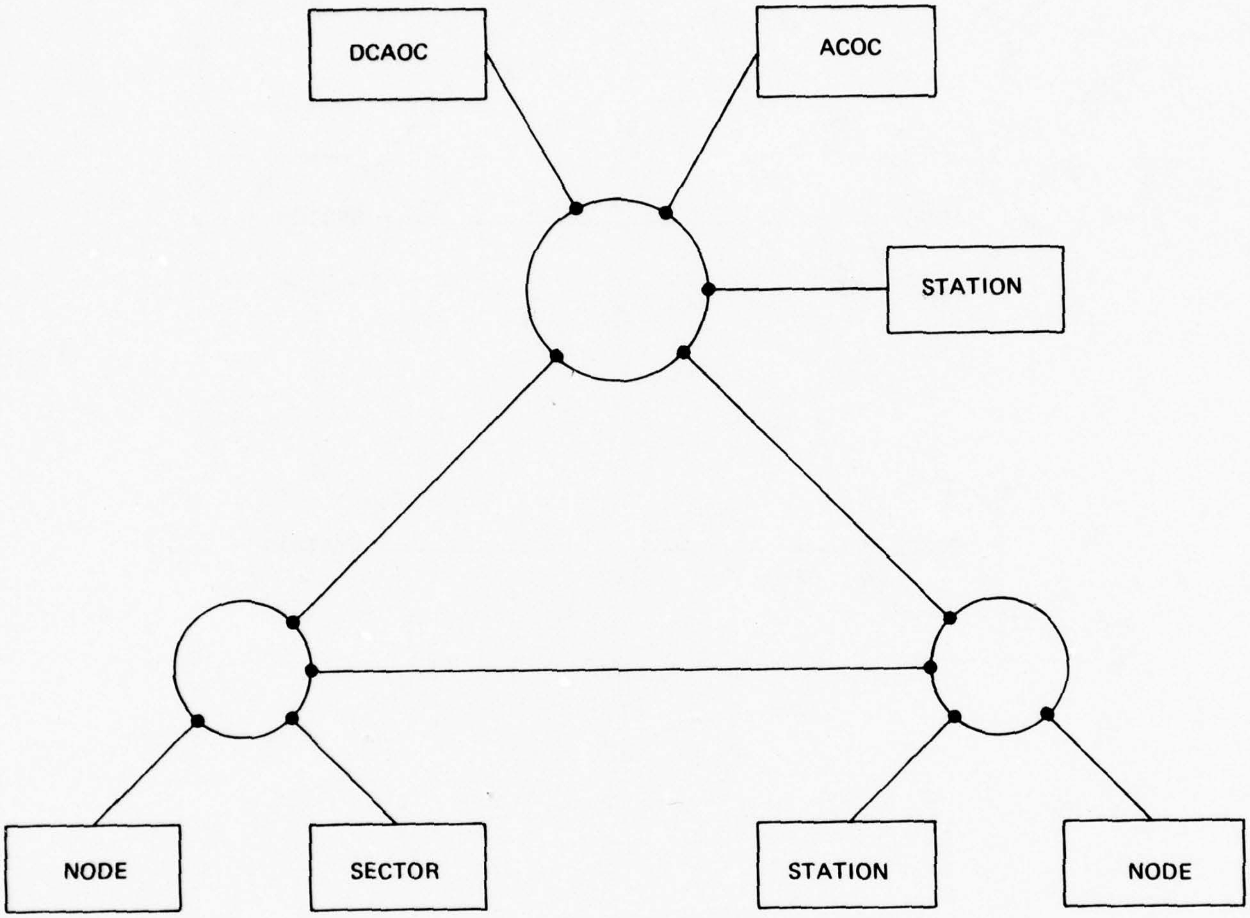


Figure 2.1-7. Multiple System Control levels mapped to the same loop

The DCS System Control hierarchy consists of five levels (DCAOC, ACOC, Sector, Node, Station) as illustrated in Figure 2.1-5. Hierarchical networks usually exhibit poor reliability since a node failure at a specific level often disconnects lower level nodes which are attached to the failed node. For good reliability a node should have two-connectivity to two different nodes in the next higher level. In the DCS network this could be accomplished via order wires. The ESM multi-loop network points out the advantages of a distributed system. The five levels could correspond to five loops connected via gateways. Different levels could also be mapped to a single loop for the case where equipment for different levels is colocated (e.g., DCA headquarters). Figure 2.1-6 illustrates a multiple loop system for the DCS hierarchy in which individual System Control levels are mapped to loops. Figure 2.1-7 illustrates a multiple loop system where some System Control levels are mapped to the same loop.

2.1.1.1 Experimental Modeling Approach

The ESM can be used to simulate failures in various System Control architectures which are being modeled. Node or link failures may be simulated by removing various ESM equipments. Link failures may be simulated by disconnecting gateway cables or removing gateway node external interface cards. Node failure may be simulated by powering down processors or removing interface cards. The original ESM system contains panel switches for putting CIE microprocessors in a don't execute state. Monitor switches can also be used to disable certain microprocessors or simulate failures.

The experimental modeling approach can be used to develop and test reporting procedures, fault detection and isolation, and fault correction. Various nodes can be loaded with special software which is used to generate faults or test detection and correction algorithms.

2.1.1.2 Mathematical Modeling Approach

Mathematical approaches for determining network reliability are given in References (2-1) to (2-14). Reference (2-2) summarizes various approaches to mathematical modeling of networks. Some of these approaches are listed below.

The connection probability of a network was studied by Kel'mans (2-8). In this study, it was assumed that nodes were perfectly reliable and all links failed independently with probability p. The probability that the graph G of b edges and n nodes is a connected graph is given by:

$$R_p(G) = \sum_{i=m}^{b} A_i \ (1-p)^i \ p^{b-i} \qquad (2\text{-}1)$$

or

$$R_p(G) = 1 - \sum_{i=d}^{b} B_i \ p^i (1-p)^{b-i} \qquad (2\text{-}2)$$

where A_i denotes the number of connected spanning subgraphs of G consisting of exactly i edges and B_i denotes the number of disconnected spanning subgraphs containing b-i edges.

The node connectivity $C_{ij}{}^n$ is the minimum number of nodes in any i-j cut. The note connectivity or the connectivity of G is denoted by

$$C^n(G) = \mathrm{Min}_{i,j} \ (C_{ij}{}^n(G)) \qquad (2\text{-}3)$$

which is the minimum number of nodes whose removal disconnects the graph. For the case where communication links are perfectly reliable and nodes are likely to fail, network reliability has been investigated by Frank (2-9). If all nodes fail independently with the same probability q, the connection probability $R_p(G)$ for a graph of *n nodes* and *node* connectivity $C^n = \omega$ can be expressed as:

$$R_p(G)=1 - \sum_{i=\omega}^{n} N_i q^i (1-q)^{n-i} \qquad (2\text{-}4)$$

where N_i denotes the number of disconnected subgraphs of G resulting from the removal of exactly i nodes.

Deterministic measures based on node-pair failure probabilities are given by Wilkov (2-10). In this study, it was assumed that all communication link failures and computer center breakdowns are statistically independent and that each communication link fails with probability p and each computer center goes down with probability q. For an n-node computer network with b communication links, it follows from the two network studies of Moore and Shannon (2-11) and Williams (2-12) that the probability Pc (a,b) of successful communication between any pair of operating nodes a and b is approximately given by:

$$Pc(a,b)=\sum_{i=0}^{b} Aa,b^e(i)(1-p)^i \, p^{b-i}, \; p \gg q \qquad (2\text{-}5)$$

and

$$Pc(a,b)=\sum_{i=0}^{n-2} Aa,b^n(i)(1-q)^i q^{n-2-i}, \; q \gg p \qquad (2\text{-}6)$$

where $Aa,b^n(i)$ is the number of combinations of i nodes such that if they are operative and the remaining n-2-i nodes fail, there is at least one communication path between nodes a and b. Aa,b^e is defined in a similar way for edges. The probability P_f (a,b) of a communication failure between any pair of operative nodes a and b is approximately given by

$$P_f(a,b) = \sum_{i=0}^{b} C_{a,b}^{e}(i)\ p^{i}(1-p)^{b-i} \quad p \gg q \qquad (2\text{-}7)$$

and

$$P_f(a,b) = \sum_{i=0}^{n-2} C_{a,b}^{n}(i) q^{i}(1-q)^{n-2-1}, \quad q \gg p, \qquad (2\text{-}8)$$

where $C_{a,b}^{e}(i)$ and $C_{a,b}^{n}(i)$ denote the number of combinations of i edges (nodes) such that the removal of only these edges (nodes) from the graph destroys all paths between nodes a and b.

A decomposition procedure for the calculation of $P_c(a,b)$ has been suggested by Moskowitz (2-13). Noting that every network consists of some conbination of series, parallel, and bridge subnetworks, it was suggested that as a first step all series and parallel links be combined. If the k links, b_1, b_2, ..., b_k are connected in and link B_i has a failure probability of P_i then the failure probability P' for the series combination is given by:

$$p' = 1 - \prod_{i}^{k} (1-p_i) \qquad (2\text{-}9)$$

assuming all links have statistically independent failure probabilities. For k links as connected in parallel, the combination would fail with a probability p' given by:

$$P' = \prod_{i} p_i . \qquad (2\text{-}10)$$

Hence, k links connected either in series or in parallel could be replaced by a single link whose reliability is given by the reliability of the combination. It has been shown by Moskowitz (2-13) and Mine (2-14) that:

$$Pc(a,b) = P_j \left\{ Pc\ (a,b) \right\}_{p_j=1} + (1-p_j) \left\{ Pc(a,b) \right\}_{P_{j=0}}, \qquad (2\text{-}11)$$

where p_j is the probability of failure for the jth link in the network and $(PC(a,b))_{pj=1}$ denotes the probability of successful communication between nodes a and b assuming that link j failes.

2.1.1.3 Simulation Modeling Approach

The reliability of a computer network may be simulated using a software simulator such as the Burroughs Operational Systems Simulator (BOSS).

BOSS is a general-purpose, discrete-events simulator program that constitutes a computerized tool for simulating the operation of a system or process. Block-diagram oriented and data-base driven, BOSS is particularly easy for the systems analyst to use. Its attractive features include:

- No formal language programming is needed for BOSS modeling. Modeling is simplified because of certain inherent biases and default characteristics in logical flow and queue servicing.

- Model parameters mapped on a BOSS block diagram may be transferred directly to input data files.

- A large library of data base error messages and notes facilitate model debugging.

- BOSS generates pertinent output reports without the need for input commands.

Unlike other general purpose simulation languages (e.g., GPSS, (SIMSCRIPT), the time required for BOSS coding is insignificant. In practice, the modeler maps BOSS parameters onto a logical flow chart which is transcribed to a file-oriented data base directly from the flow chart to the file. The series of input files comprises data input to be executed with BOSS object machine codes.

Failures on links and nodes may be simulated on a random basis using Poisson distributors. The network reliability can then be determined and tabulated.

2.1.2 Distributed System Advantages

A distributed system implementation for the ESM simulation facility has certain advantages over a hierachical or centralized system, or a simulator developed strictly in software. A distributed system such as the ESM where nodes configured around microcomputers programmable in a higher level language provides a powerful simulation capability. The nodes can simulate processing elements, and the interprocessor communication network and inter loop links can simulate telephone line or other connections in a computer network. In addition to its simulation capability, the ESM provides a more powerful learning and demonstration tool than a simulator implemented completely in software.

The ESM multiloop network with its indirect method of addressing process names implemented via nodal LID/FAD conversion table provides an efficient method for message communication between any two nodes in the network. Using such a scheme a logically defined network can be implemented. Thus, the SYSCON hierarchy can be logically defined onto a multiloop system with the reliability advantages of a distributed system rather than a physically defined hierarchical system.

The distributed ESM network has many reliability advantages over a centralized system. The network operates in a degraded mode of operation. For example, since the five loop ESM network exhibits two-connectivity, devices can continue to talk to each other with the loss of a link connecting loops. The distributed data base allows single-host failures. The distributed file directories are duplicated on both PDP-11's and the data is duplicated on the B776 processor. Thus, a loss of a single processor does not affect the operation of the distributed data base. The ESM distributes control functions so that a single element failure cannot bring down the system. An example of this would be write token regeneration for the case of write token loss. Each node in the loop has the ability to regenerate a write token with each using a multiple of the time out parameter. Software maintenance and reliability is better for the distributed ESM since programs are small and dedicated to a simple task depending on the nodal interface. Programs are of a size less than the 32k bytes of nodal memory and much of the software is duplicated at the nodes (e.g., I/O queue maintenance).

The distributed system has an advantage in throughput over a centralized system. In a centralized system the throughput is limited by the speed of a central supervisor whereas in the distributed system multicomputing can be utilized where interface controllers are configured around individual microprocessors which operate in parallel. The distributed system is more adaptable/ flexible since nodes may be used for different functions depending on the software loaded. The distributed system is low in cost. The nodes of the network consist of inexpensive microprocessors and the links consist of twisted pair wire. The entire network can be contained within a building and gateway connections can be made to networks and equipment outside of the building.

2.1.3 Blocking and Deadlock Situations

Various loop protocols can be used on a loop network. Some protocols allow only one transmission at a time, others allow more than one. Some protocols can result in a deadlock situation called "loop clogging" where all nodes in the loop are locked out or prevented from writing to the loop.

There are three main types of loop protocols currently used. The first type is a Newhall protocol which uses a special control packet called a write token. A node which processes the write token controls the loop. That node is the only node which has the authority to write to the loop and thus the protocol allows only one transmission or communication on the loop at a time. A Pierce protocol appears asa lazy susan in which a fixed size packet can be inserted into an empty slot. The destination node removes the packed from the slot. In the protocol multiple transmissions are possible on the loop. When undelivered messages are allowed to circulate indefinitely on the loop such that all the available slots are used up, the loop clogs such that nodes are prevented from writing to the loop.

A Reames protocol allows multiple transmissions and variable length packets. It is implemented by an expanding and contracting loop in which each node contains a FIFO queue. Each node can read and write at the same time, thus, incoming traffic can be diverted to the FIFO queue when a node has data to write. The Reames loop protocol is also subject to clogging.

The ESM system guarantees that clogging can never occur. The Newhall protocol can be divided into two types. The WT-1 version allows an entire nodal queue to be emptied when a write token (WT) is received. The WT-2 version allows only one packet to be written to the loop when the WT is received; the WT must be sent out onto the loop after the single packet is written. The ESM system uses the latter version with packets being up to 256 bytes in length. The ESM node performs a destructive write thus overwriting any bytes that may be circulating on the loop due to undelivered messages. When a write token is lost, it is regenerated by the nodes which have multiples of the write token timeout parameter. The loop acquisition time in the WT-2 protocol is bounded and is a known quantity used for the write token timeout parameter.

The ESM original three loop network used a directed write token. The write token was sent to each adjacent node on the loop and interpreted as a write token by the nodal software. Experiments on the ESM system where nodes were put in a don't execute state forcing WT regeneration and misdirected WT's where two would be generated and eventually quenched due to the destructive write capability, indicated that the system was extremely reliable, fail-soft, and not likely to deadlock.

Any loss of synchronization in a node is automatically resynchronized by a node's hardware in a time span of not more than nine fields. A temporary form of node paralysis can occur when the end of packet character is distorted by noise. The NCU BEX2 command that causes the node to read will cause the node to hang up waiting for a new character which is not forthcoming. The next write token that is directed to the node will supply the required end of packet indication at the expense of the loss of the write token. After a timeout, some node in the loop will generate a new write token. In any case, a sufficiently long hang up of the NCU which sets the program counter of the NCU to 0. This restarts the NCU in the "wait" state until resynchronization occurs. The CIE then restarts the NCU in the regular fashion.

The ESMD loop 4 uses an orbiting WT rather than a directed WT. Under no load conditions the loop has a single WT circulating at the basic loop data rate. When a node wishes to write a packet, it preloads the packet into the primary output buffer and a WT into the secondary output buffer. It then modifies the address comparison memory (ACM) on the LIU so that the WT address (e.g., 255) is recognized by the node. When the WT is recognized by the LIU it starts writing the output buffers to the loop and generates an interrupt to the CIP. If there are additional packets waiting in the output queue, the CIP loads the output buffers. If there are no additional packets to be written, the CIP notifies the ACM that the WT is no longer recognized.

Various schemes can also be used to counter clogging in the Pierce and Reames protocol. In the Burroughs ADO loop which uses a Pierce protocol, a loop monitor node detects and corrects clogging. In the Reames protocol additional bits are contained in the packet which nodes can modify and examine in order to remove circulating messages.

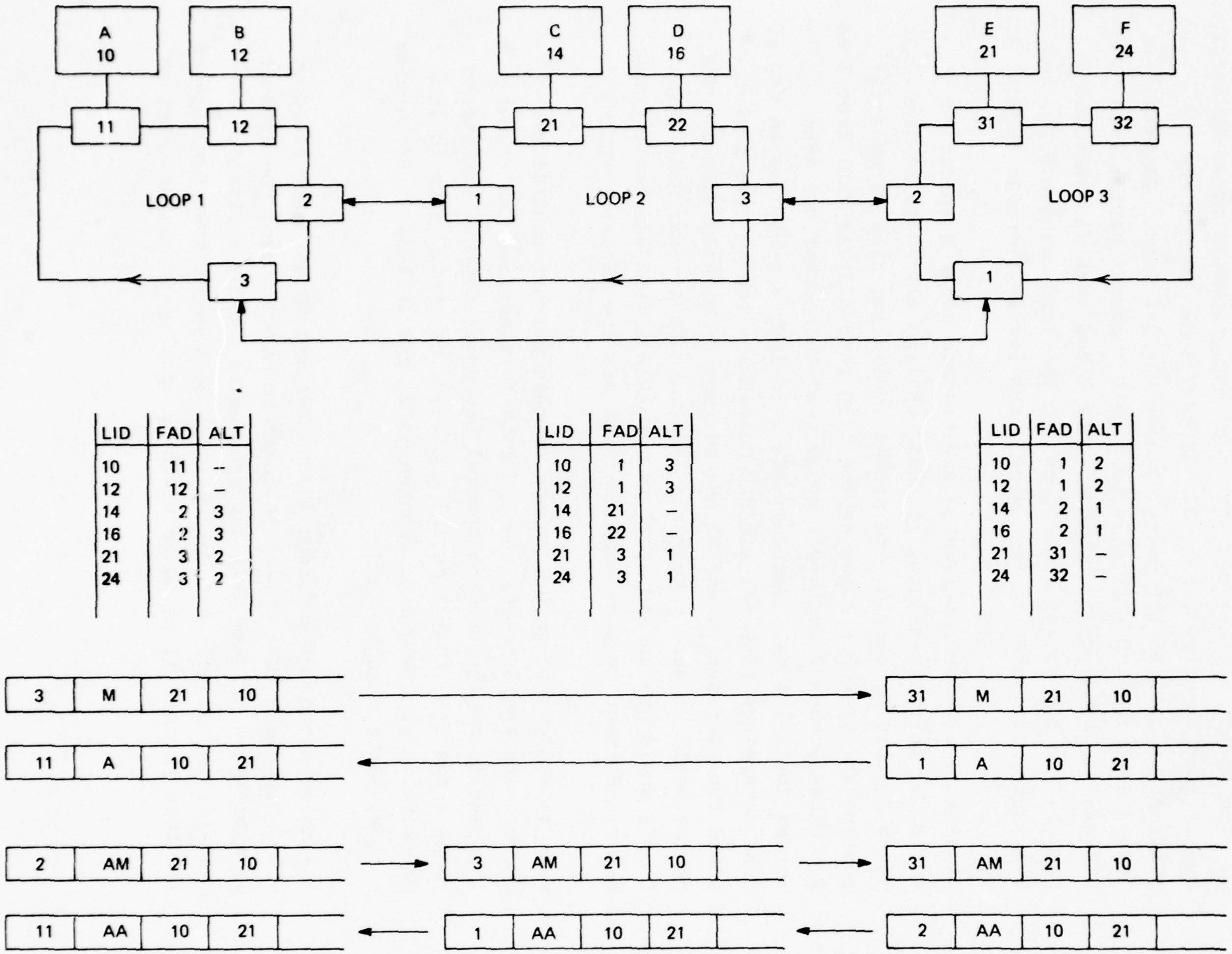
Another area where deadlock situations can occur is the host to host bid interprocess communication language. A possible deadlock situation is found in Section 2.2.8. Access commands are used to define the interprocessor communication language. A deadlock situation can occur when files are being transferred between two machines. As each record of the file is sent, a response is sent by the receiver saying that it had received the lost record and to send the next record. This handshaking is necessary since records may be destined to devices of various speeds (e.g., disk, tape, printer). Deadlocks are prevented in the program by means of time outs. The subroutine module that reads from the loop (FRDLP) has a parameter (ITM) that can be set when a host response is expected. The subroutine will only wait 10 seconds for a response from the other host processor. If no input is received within that time control returns to the main program and the outstanding file is closed and an error message is sent to the ESM terminal that requested the file transfer indicating that there is no response from the remote processor.

2.1.4 Routing Methods

When a link goes down a routing method must be employed in order to avoid the bad link. In the three loop ESM system, its two connectivity feature implies that a node can talk to any other node via an alternate path when the primary path link is down. Each ESM loop has at least two gateway nodes. Alternate routing is very simple to implement the ESM system with its logical ID (LID)/functional address (FAD) conversion tables. A node uses the FAD found in its table at a location (index) equal to the LID as the loop address when sending the packet. It may reach its final destination via other loops. If the destination is correctly reached (i.e., a good LPC for the original three ESM loops, a good CRC for loops 4 and 5), a CIP to CIP ACK message is sent back to the originating node.

If a NAK is received or a timeout period is exceeded with no response, the message would be retransmitted. After a specified number of retries, alternate routing would be utilized by using the loop address of another gateway node in the loop. By conversion, gateway nodes use an FAD or read address equal to the loop number to which they send messages.

An example of how alternate routing is implemented with a multiloop architecture using indirect addressing is given in Figure 2.1-8. Let us assume that host processor A on loop 1 wishes to send a message to system process 21. Host A need not know where process 21 resides in the network. Let us assume for the example that process 21 resides on host processor E in loop 3. Host A sends a packet to its CIP with 21 as the destination LID and 10 as its source LID. The CIP looks in its LID/FAD conversion table and formats a packet using an FAD or loop address equal to



LID	FAD	ALT
10	11	--
12	12	–
14	2	3
16	2	3
21	3	2
24	3	2

LID	FAD	ALT
10	1	3
12	1	3
14	21	–
16	22	–
21	3	1
24	3	1

LID	FAD	ALT
10	1	2
12	1	2
14	2	1
16	2	1
21	31	–
24	32	–

Figure 2.1-8. Indirect Method of Addressing

3. The packet is sent out onto the loop, bypasses nodes 12 and 2 and is read by gateway node 3. Gateway node 3 sends the information part of the packet across the 1-3 link. Gateway 1 in loop 3 uses its LID/FAD table to format a packet having loop address 31. The packet bypasses node 2 and node 31 reads the packet. An ACK packet is sent out on the loop using LID 10, and the packet is linked to the input queue for deliverence to Host A.

If node 11 had not received an ACK message after a specified number of retransmissions, it would utilize alternate routing. It would do this by marking the packet indicating that alternate routing was used and changing the loop read address(FAD) from 3 to 2. Gateway node 2 in loop 1 would read the packet and send it across the 1-2 link. Gateway node 1 in loop 2 would use an FAD of 3 as determined from its LID/FAD conversion table. The packet would bypass nodes 21 and 22 and be read by gateway node 3. The packet would be sent across the 2-3 link and gateway node 2 in loop 3 would use an FAD of 31 as determined from its table. The acknowledgement message would be sent via the alterate route.

Node 11 would also report to one or more network control processors who could remove the 1-3 link from service for repair. This would involve sending special broadcast control packets to loops 1 and 3 so that link 1-3 would not be used. Thus in loop 1 FAD entries of 3 would be changed to 2, and in loop 3 FAD entries of 1 would be changed to 2.

The above method of indirect addressing can be used for resource allocation such that processes could be moved around the network so that spare or less utilized processors can be utilized. For example, let us say that Host E is to be brought down for service and thus process 21 is to be moved to another processor. Let us say that it is determined (possibly by some bid-quotation scheme) that host D of loop 2 is to handle process 21. In order to move the process, control packets would be broadcast in each loop to change the LID/FAD tables. In loop 1 the FAD for LID location 21 would be changed from 3 to 2 (ALT would now be 3), in loop 2 the FAD entry would be changed from 3 to 22, and in loop 3 the FAD entry would be changed from 31 to 2 (ALT would be 1).

2.2 Identification of Adverse Factors

2.2.1 Introduction

This section summarizes the results of the study of adverse factors on system reliability (task 2.2 of the ESMD proposal). The study is primarily concerned with the ESM architecture consisting of four interconnected loop networks implemented by microprocessors. Solutions to system reliability adverse factors are presented in terms of ESM hardware and software although they are also applicable to other systems. Section 2.3 summarizes the specific fail-soft features of the ESM.

The adverse factors identified and discussed below are node failures, interprocessor communication network failures, link failures, file corruption, and user language considerations with respect to logical ID/functional address conversion table modifications.

2.2.2 Node Failures

An ESM node functional diagram is given in Figure 2.2-1. A failure of the loop interface unit (LIU) or control and interface processor (CIP-BDS) can produce a node failure. A memory failure may or may not be serious enough to cause a node failure depending on byte location. Memory failures at locations containing program are more likely to be serious failures than failures at locations containing data. An external interface card failure need not be serious unless the external interface is a control processor (e.g., data base processor, system control processor, security monitor). It has been estimated that the mean time between failures (MTBF) for the LIU is 22000 hours and that for a BDS microprocessor with memory is 10000 hours giving a combined value of 7000 hours.

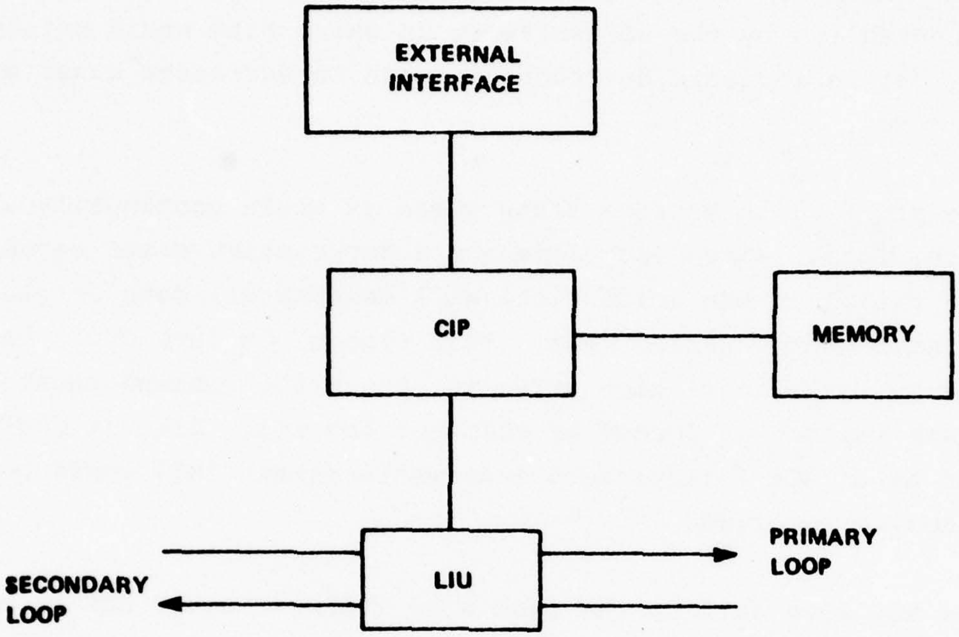


Figure 2.2-1. Communication Loop Node

A node may fail in a read state and continually remove data (which may not normally be addressed to it) from the loop. This type of failure could be caused by a bad address comparison memory (ACM) chip controlled by the BDS software in which bits would get incorrectly set to indicate destructive reads on addresses owned by other nodes.

A node may fail in a write state where it would continually write onto the loop. Since each node has a destructive write capability a node failed in the write state will destroy all data originating upstream from the failed node. This type of failure could be caused by a continual high value for the write command caused by hardware failure or forced by software control. Also it could be caused by an ACM failure such that write tokens (WT) would be continually received.

A node may also fail in the pass mode where it would act as a delayed repeater to the data stream and would neither be able to read data or write data. This type of failure is not as serious as the above two since the failure is local and there is no effect on the other nodes in the system since the data stream is not modified. The system may be affected if the external device connected to the failed node is a host processor with control responsibility which could not communicate with other host processors.

Failures in the read or write mode are serious in that they may interrupt the data flow and affect system operation. The solution to preventing these failures from corrupting the system is to remove these failed nodes from the loop network by means of the loop-back capability described in Section 2.2.4. It should also be noted that since the ACM is controlled by BDS software it is possible that a malicious programmer could simulate a hardware failure in the read or write mode. For this situation, the ability to remove a node from the loop via the loop-back mechanism is necessary (applies to loop 4 only).

The loss of certain host processors may affect the operation of the system. For example, Mode 3 of the ESM User Language (System Control) is contained on only one host processor (PDP11 B, Loop 2). In an operational system the loss of this processor would mean the loss of Mode 3 unless the function could be moved to another processor. This could be controlled by a monitor node who could send a control message to another processor to start handling mode 3 since a report was received that the previous node 3 handler processor does not respond.

A desirable feature for reporting the unavailability of external equipment due to failure or power-down is an Acking scheme across the CIP-external device interface. The CIP could periodically test to see if its external equipment is properly working. If there is no response, the CIP could report the unavailability of its external equipment to a resource manager associated with a node or set of nodes. The unavailability of the external equipment could then be displayed for system users and critical functions for the case of unavailable processors could be reallocated to other available processors.

An example of host processor failure that exists in the current ESM is the automatic connection of a terminal to an alternate dialogue director. This experiment is performed by putting the HST1 node in a don't execute, clear state via switches in cabinet #1. CRT8 in loop 3, whose microcode (object file CRT8S.OBJ) uses HST1 (processor A loop 1) as a primary dialogue director, never receives an ACK from HST1. It then attempts alternate routing and still receives no ACK; so it sends the dialogue input to HST5, loop 2 which is connected to host processor B.

The experiment as it is currently performed has some shortcomings. It actually simulates a node failure rather than a processor failure since the CIP generates an ACK and there is no CIP-HOST Acking scheme. Thus the current experiment would not result in terminal reattachment if the processor was powered down and the CIP was in an execute state. This again points out the need for a CIP-external device Acking scheme; i.e., it would be desirable for the CIP to be able to determine the availabilty of the external equipment it is connected to, for the case of host processors an occasional ARE YOU THERE?, YES I AM exchange would suffice. Also in the case of the existing experiment, CRT8 does not learn to adapt to the inavailability of HST1. With each transmission CRT8 attempts to use HST1, attempts alternate routing, and then finally transmits to HST5. The attachment of CRT8 to HST5 in this situation must be done by operator intervention (Mode 3 on processor B) to chage the LID/FAD conversion table of CRT8 and GAT2-3 (i.e., change LID 1 to FAD 2). The loop utility (LPFT) described in Section 2.2.8 along with new versions of the microcode for CRT4, CRT8 provide a dynamic attach facility which changes the processor that a terminal is connected to using a new control packet (utilizing bit 2 of header control byte D3).

Error recovery procedures must be employed to detect bad nodes, reconfigure the network to isolate the bad node, report the failure to system users, and reallocate functions for the new network configuration. An error recovery procedure for detecting a node that has failed in the write state resulting in a noise-producing node is described below.

Detection of such nodes is relatively simple because of the sequential nature of the passage of information. For example, in a WT-1 or WT-2 loop where write tokens are regenerated after time-out, the node just downstream of the noisy node will be the only node that never receives a write-token. It can perform its own wrap-around and signal the node upstream of the noisy node to do the same. This will isolate the noisy node. Another method that more generally applies is the use of framing character detectors. A noisy node will generally produce fewer framing characters than normal or else much too many. If any node detects such a condition over a period of time, it intercepts the bit stream and acts as a controller to search out the noisy node.

Once a faulty node is detected and isolated from the network, diagnostics must be run in order to determine which card is bad. Certain failures in the processor, LIU or certain locations of the memory cards may prohibit the running or loading of diagnostics. Diagnostics should be run without interrupting the operation of the newly configured system. Hot-card replacement procedures should be utilized so that the loop cabinet need not be powered down. Certain LIU functions must be tested using two nodes where one node may operate as a writer and the other node as a reader.

2.2.3 Interprocessor Communication Network Failure

The interprocessor communication network must degrade gracefuly. The network for ESM is the twisted pair cable that connects the nodes in a loop or ring configuration. The loops in ESM are completely contained within cabinets thus there is very little chance that the backplane resident communication network can be broken.

In the general case, loops or other architectures can be distributed rather than localized within a cabinet. The distributed approach would be used for extended reach applications in which each node would reside near the external equipment and would have its own power supply. In this situation it is possible for the cable to become broken. This is particularly likely in military applications where battle damage may cut the interprocessor communications network. Unlike many bus architectures where a cut cable will bring down the entire system, the ESM loopback capability provides for automatic reconfiguration when the interconnecting cable is cut. In the ESM system battle damaged areas of the network can be removed and communication can continue along the surviving reconfigured network. The automatic loopback scheme described below reconfigures a loop network to isolate adjacent node failures.

2.2.5 Link Failures

In the ESM system links are used to connect loops. The transmission media is ribbon cable although it is meant to simulate data communication lines which would connect loops at different locations. Link interfacing is done via gateway nodes. Multiple loop networks may be distributed geographically where each loop would be contained within a building, or functionally where each loop may handle a set of functions and multicomputing loops would be colocated and communicate via high speed direct connections.

Since a gateway node is software controlled and contains considerable queue space, it provides loop independence or independence from other types of networks (e.g., AUTODIN II, TCCF, SDLC). This independence may be in the areas of protocols, data rates, and security.

The link connection scheme for the original three loop ESM system is given in Figure 2.2-3. The system is fully connected since each loop has a direct link to the other two loops. Loop independence was demonstrated in ESM by running the loops at three different loop rates (switch selectable) and demonstrating that loop to loop communication continued.

Since there are two gateways per loop in the original three loop ESM configuration, alternate routing is possible. A demonstration of this begins with the system configured so that CRT8 (loop 3) is attached to HST1 (loop 1). The link between loops 1 and 3 is broken by removing a gateway interface card from cabinet 1. CRT8 now attempts to communicate with loop 1 but doesn't receive an ACK. It then sends the packet to loop 2. In loop 2 the packet is sent to the gateway node connected to loop 1. In loop 1 the packet is sent to HST1 and the ACK message is sent back to CRT8 via the alternate route. Thus there is always a primary route via the gateway that connects directly to the destination loop and an alternate route via the gateway that connects to an intermediate

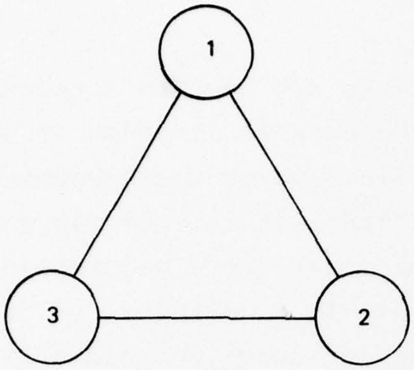


Figure 2.2-3. Original ESM configuration

loop. For the demonstration the link is removed from service by modifying LID/FAD conversion tables. CRT8's table is changed so that LID 1 (for host 1) is mapped to FAD 2 (for gateway 3-2), and HST1's table is changed so that LID8 (for CRT8) is mapped to FAD2 (for gateway 1-2).

The network configuration is defined by the nodal LID/FAD conversion tables. The tables can be changed by control packets that are interpreted by the nodes. In this manner dynamic network reconfigurations are possible. As links fail table changes can be made so that alternate routing via an intermediate loop can be used.

The addition of the fourth loop will result in the configuration of Figure 2.2-4. Note that loop 3 will have three gateway nodes and loop 4 will have one ESM connected gateway thus alternate routing cannot be done via loop 4. However, the addition of the Feasibility Development Model (FDM-Loop 5) of the Modular System Control Development Model contract will supply additional flexibility. Figure 2.2-5 shows two anticipated configurations for the five loop network. Configuration a.) is superior for handling link failures since each loop has at least two gateway nodes (i.e., the network has two-connectivity) and alternate routing via intermediate loops can be utilized.

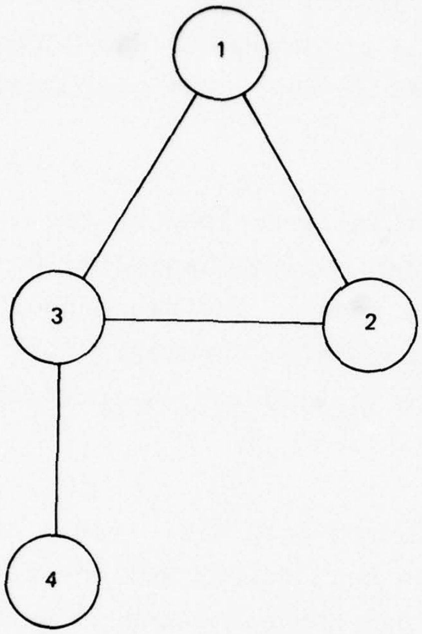


Figure 2.2-4 Four loop configuration

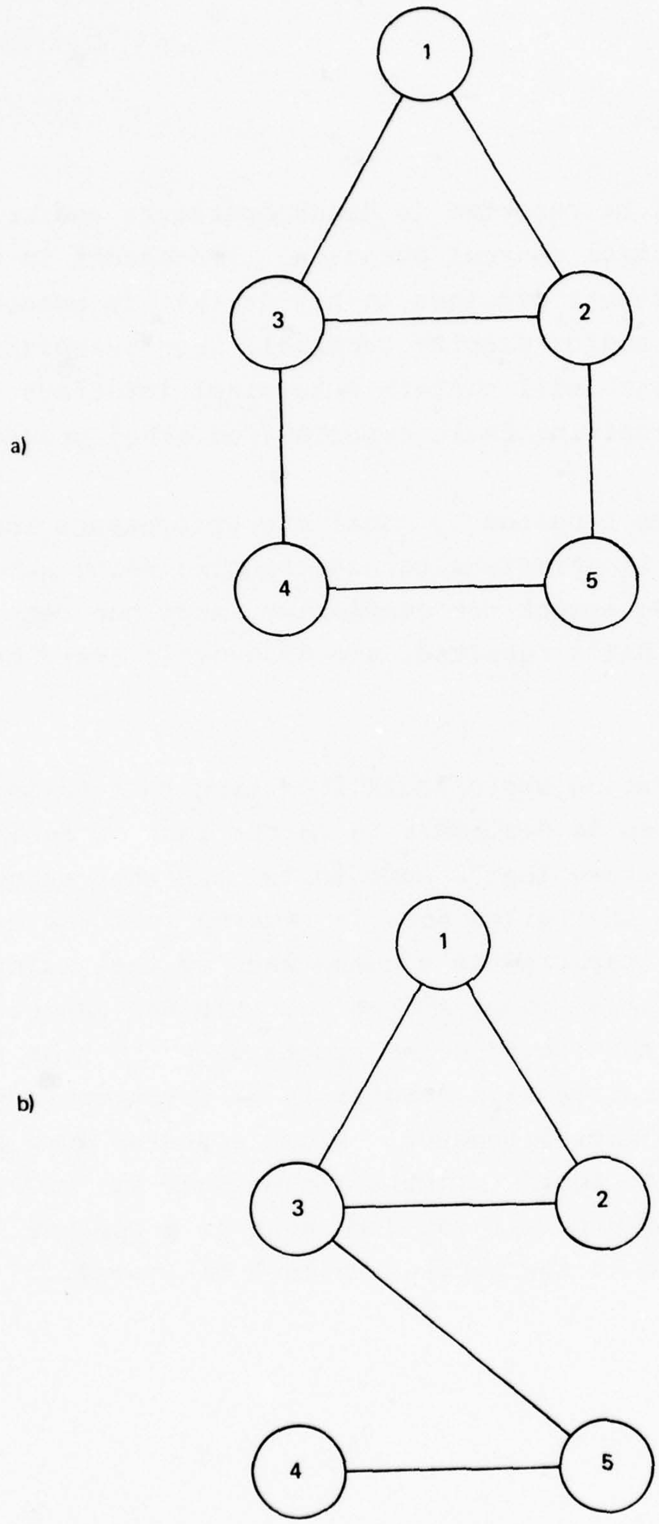


Figure 2.2-5. Five loop configuration

2.2.6 Fault Reporting

System faults should be reported to human operators and/or users for informational and/or control purposes. Processors in distributed nodes should report problems to a node that is connected to a control processor and/or display terminal. For example the ESM node G of loop 4, which will contain a terminal interface card, could be used for receiving fault reports from other processors.

Faults which could be reported by nodal microprocessors include automatic loop-back in effect, alternate routing being used, external interface equipment non responsive, node non responsive, excessive number of NAK's received, and diagnostic tests being run.

An example of a situation where fault reporting to a human for informational purposes is desirable is in the case of automatic loop-back. Let us assume that a node fails such that automatic loop-back occurs and the failed node is removed from the network. This failure must be reported to a human user so that maintainance can be done on the failed node, and so that LID/FAD tables can be modified to reflect the reconfigured system with the node removed. In the ESM, automatic loop-back results in an interrupt being generated to the BDS microprocessor. After a status word is examined the processor could format a packet such as "LOOP-BACK IN EFFECT - NODE 13". The packet could be sent to a special LID which would be routed to the terminal node G in loop 4.

2.2.7 File Corruption

Another adverse factor is the corruption of files in the system. Files may become corrupted due to disk failures or not available due to processor failure for the processor that controls the disk. Certain key files such as the system tables may become corrupted (e.g., INFO.DAT which contains ESM LID/FAD conversion table images).

The effect of file corruption can be minimized by introducing multiple copies of files on different disks. It is recommended that the ESM distributed data base contain multiple copies of both directories and data. Distributed directories are used for associating an LID with a record of a file to indicate where it resides in the system. Directories are duplicated on both ESM PDP-11's. Data files should also be duplicated; the loop 4 B776 processor can be used to hold a duplicate copy of the data files distributed on the PDP-11 processors.

For fail-soft operation with respect to file corruption, a system utility is necessary in order for duplicate files to be created on different processor disks in the system. A loop utility program which creates multiple copies of 80 byte files is described below.

2.2.8 Loop Utility

A utility has been written for the ESM (LPFT) which improves the failsoft operation of the system. The utility allows terminals to be reattached to another processor, allows files to be sent to another processor's disk, printer, or tape, and allows files to be sent from another machine's disk. A flow diagram for the machine-human dialogue for LPFT is given in Figure 2.2-6.

A host-to-host level interprocessor communication and synchronization scheme is implemented in FORTRAN for LPFT by means of access commands contained in byte 7 (D7) of the packet. These access commands are given in Table 2.2-1.

Table 2.2-1. LPFT Host-To-Host Access Commands

Access Command	D7
Request file X Record size Y	20
Write File X Record Size Y to Device Z	21
Record to Write	22
End of File	23
Record Received - Send Next	24

The terminal reconnection facility is provided by a new control packet generated by the dialogue director host and interpeted by the CRT node. The control packet has bit 2 of byte D3 on with the new dialogue director LID contained in byte D7. A new version of CRT4.OBJ and CRT8.OBJ is supplied to interpret this new control packet.

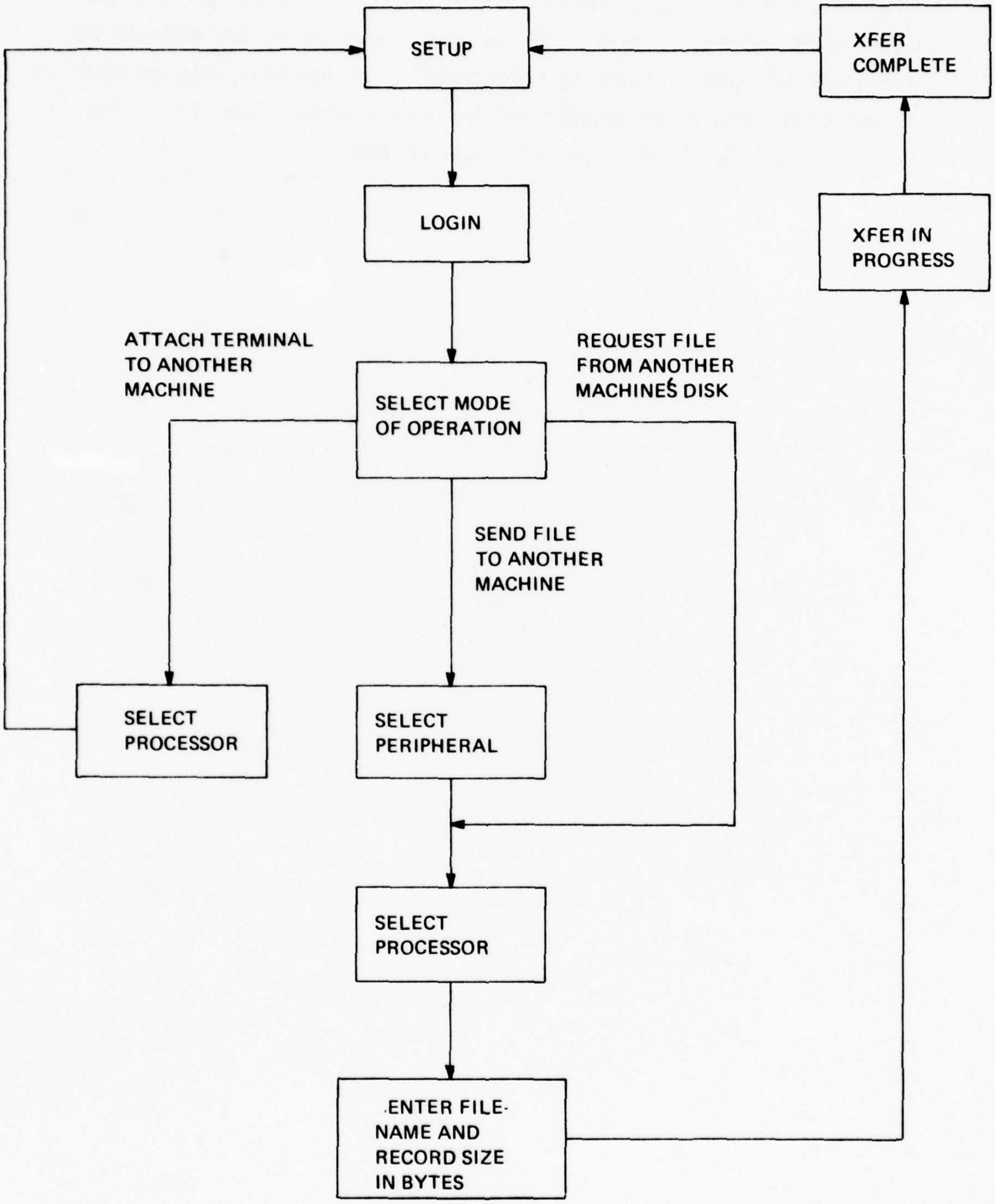

SETUP
XFER COMPLETE
LOGIN
XFER IN PROGRESS
ATTACH TERMINAL TO ANOTHER MACHINE
REQUEST FILE FROM ANOTHER MACHINE'S DISK
SELECT MODE OF OPERATION
SEND FILE TO ANOTHER MACHINE
SELECT PROCESSOR
SELECT PERIPHERAL
SELECT PROCESSOR
ENTER FILE-NAME AND RECORD SIZE IN BYTES

Figure 2.2-6 LPFT Flow Diagram

LPFT allows multiple copies of files to be made on another machine's disk or tape, allows peripherals to be shared by processors, and allows ESM terminals to dynamically attach to other processors as commanded by the system user in order to improve the fail-soft capability of ESM.

2.2.9 User Language Considerations

System Control fail-soft experiments are performed via the ESM User Language. LID/FAD table updates and read addresses can be changed in the System Control Mode 3. Changes are entered on ESM terminals and the dialogue director host processor (only host B implements Mode 3) formats control packets that are interpreted by the node whose data memory is modified. Modifications are stored on the host computer's disk (file INFO.DAT) and displays of nodal memory parameters are presented via System Inquiry Mode 2 of the User Language.

The system file should be stored on only one host processor's disk and modifications should only be done by that dialogue director host. Thus Modes 2 and 3 of the User Language are only resident on one host. A system lock should occur after a change has been made so that only one change can be made at a time. Checking and verification of inputs should be used for Mode 3 so that minor typing changes can be detected before nodal memories are modified incorrectly.

The file INFO.DAT keeps track of changes as they are made to the system. A baseline system that corresponds to the table parameters contained in the nodal microcode is contained in the file INFOPM.DAT. At system startup time or when the system is cleared the PDP-11 command file STESM deletes old versions of INFO.DAT. System modifications are then made to the file INFO.DAT which is used to generate the displays of Mode 2.

Although INFO.DAT is only stored on one processor, it may be useful to store INFOPM.DAT (corresponding to the baseline system) on all system dialogue director hosts so that Mode 2 can be accessed on all processors. If the current tables wish to be viewed (i.e., INFO.DAT in Mode 2 on Host B), the user can exit from the User Language and attach his terminal to Host processor B using LPFT.

2.3 ESM Fail-Soft Features

This section describes the ESM features that will be used to counter the adverse factors described above. This section summarizes the work done for Task 2.3 of the ESMD proposal. Loop 4 of ESM will be implemented using the double loop architecture with the automatic loop-back feature described in Section 2.2.4. Task 2.3 will provide inputs to the acceptance test definition of Task 4.7.

The following demonstrations will be done to simulate SYSCON failures and indicate fail-soft operation:

i.) Node Failure Simulation - Remove an LIU card from Loop 4. Automatic loop-back occurs on both sides of the removed node. The fault is reported to the CRT terminal attached to node G. Demonstrate that terminal-host communication is still possible with newly configured system and recovery to original configuration after "bad" node is repaired.

ii.) Interprocessor Communication Network Failure Simulation - Ground or open a primary loop wire. Automatic loop-back occurs. Report fault to CRT terminal node. Demonstrate terminal-host communication still possible.

iii.) Host Processor Failure Simulation - Power down the B776 to simulate a failure. The BDS connected to the B776 occasionally sends an ARE YOU THERE? message. When the B776 is non-responsive, the BDS sends a fault report message to the terminal node.

iv.) Link Failure Simulation - Remove a gateway interface card connecting ESMD (loop 4) to ESM. Attempt to send a message from loop 4 to ESM; when no ACK is received report fault to terminal node. Alternate routing was demonstrated for the original three ESM loops.

v.) Loop Utility Demonstration (LPFT) - This program is described in Section 2.2.8. The features of file transfer and resulting duplication on other processors, peripheral sharing, and terminal reconnection for achieving fail-soft operation are demonstrated.

REFERENCES

(2-1) C. Kisboa, P. M. Lin, B. J. Leon, "A New Recursive Approach to the Reliability Analysis of Large-Scale Networks", Purdue University, TR-EE 76-25, Aug. 1976, Contract F30602-72-0438.

(2-2) R. S. Wilkov, "Analysis and Design of Reliable Computer Networks", Proc. IEEE Trans. on Communications, Vol. COM-20, No. 3, June 1972.

(2-3) E. Hafner, Z. Nenadal, "Enhancing the Availability of a Loop System by Meshing", Proc. 1976 Int. Zurich Seminar.

(2-4) H. Frank, I. Frisch, Communication, Transmission, and Transportation Networks, Addison - Wesley, 1971.

(2-5) N. Abramson, F. Kuo, Computer - Communication Networks, Prentice-Hall, 1973.

(2-6) M. Schwartz, Computer - Communication Network Design and Analysis, Prentice-Hall, 1977.

(2-7) N. Spyratos, A. Bowen, "A Fast Algorithm for Reliability Calculations in Sparse Networks", Networks, 7:pp. 227-246.

(2-8) A. K. Kel'mans, "Connectivity of Probablistic Networks", Automat. Remote COntr., No. 3, pp. 444-460, 1967.

(2-9) H. Frank, "Maximally Reliable Node Weighted Draphs", Proc. 3rd Annu. Princeton Conf. Information Sciences and Systems, Man 1969, pp. 1-6.

(2-10) R. S. Wilkov, "Reliability Considerations in Computer Network Design", Proc. Int. Fed. Inform. Process. Soc. Congr. 1971, Ljubljana, Yugoslavia.

(2-11) E. Moore, C. Shannon, "Reliable Circuits Using Less Reliable Relays", J. Franklin Inst., Vo. 262, pp. 191-208, Sept. 1956.

(2-12) G. Williams, "The Design of Survivable Communication Networks", IEEE Trans. Commun. Syst., Vol. CS-11, pp. 230-247, June 1963.

(2-13) F. Moskowitz, "The Analysis of Redundancy Networks", AIEE Trans. (Commun. Electron.) Vol. 39, pp. 627-632, 1968.

(2-14) H. Mine, "Reliability of Physical Systems", IRE Trans. Circuit Theory, Vol. CT-6, pp. 138-151, May 1959.

3.1 INTRODUCTION

The ESMD security task will examine the security-related characteristics of potential Defense Communications System (DCS) SYSCON Architectures. The DCS system control subsystem is intended to provide network control, traffic control, performance assessment and status monitoring, and technical control (Sch 74). The essential feature of a system control subsystem for the integrated DCS of the 1980's is a requirement for a secure distributed processing network environment. SYSCON is described in (Sch 74) as a multi-level management hierarchy, with excessively higher levels of the hierarchy assuming more and more of the responsiblity for monitoring and controlling larger portions of the DCS (see Figure 3-1). "The first level of control will act as an arbiter in resolving interarea control decisions as well as coordinating control actions affecting more than one area. The second level will be geographically dispersed and will report to level one. The third level, normally performing the functions of data reduction analysis and formatting, interfaces with the switching and transmission nodes. It will provide a minimum degree of autonomous decision making and control. The last level, level four, will provide the necessary functions of testing, patching, adjusting, maintenance, status monitoring, and control that must be performed at the equipment level...The system control paths of communications are assumed to comprise a mix of dedicated and switched circuits with the capability of voice, teletype and processor-to-processor communications." (Sch 74)

A more recent DCS system control hierarchy as described in the Modular System Control Development Model (MSCDM) Phase I Final Report consists of five levels. These levels are: I - Worldwide Control, II - Theater Control, III - Sector Control, IV - Nodal Control and V - Station Control. Levels I and II are involved with network control, levels II and III are involved with traffic control, and levels III and IV are concerned with transmission control.

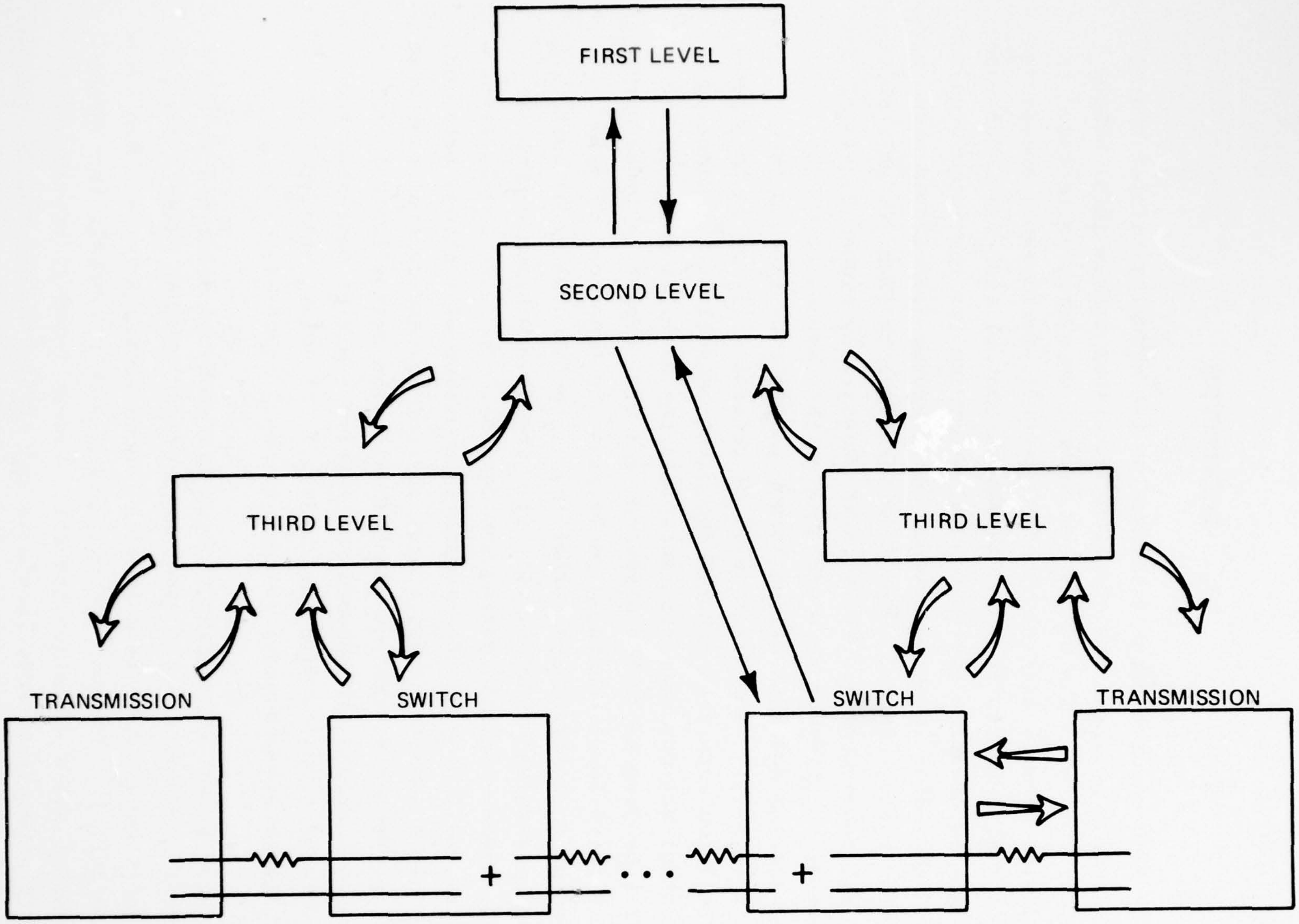


Figure 3-1. System Control and Management Hierarchy

The need for secure distributed processing in a geographically dispersed, multi-level network for system control of the DCS is becoming more and more apparent. Just what the specific security requirements of SYSCON subsystem might be is still not completely known or agreed upon. For this reason, though system control is a unique application with special requirements all its own, the intent of this task will be to assess the security requirements of distributed data processing networks in general, with particular emphasis on military security problems, and to describe the means by which they can be met.

There are several classes of distributed computer systems. Ramamoorthy (RK 76) distinguishes between (1) systems of parallel processors, (2) multi-processor configurations, (3) computer networks, and (4) his own model of a distributed computer system, which he calls the DCS. "Basically, the distinction is based on the manner in which the processing elements cooperate with each other to work on a given job or on different jobs. In parallel systems all the processors may be performing the same operation on different sets of operands, and in a multiprocessor system each processor may be working on different jobs and the degree of cooperation may be minimal. In computer networks the processors are mostly autonomous general-purpose computers that in most cases are geographically apart and are connected through a switching or a communication network. However, in a DCS (Class 4) the PEs (Processing Elements) work in a cooperative way on a set of related jobs: in other words, the processing of a given job is distributed among various PEs."

In its most general form, SYSCON is potentially a hybrid of all four classes of distributed computer systems. Taken as a whole it is essentially a computer network, but its nodes could assume any one of the other shapes of a DCS. Because of the structural characteristics available for the ESM simulation studies, this task will restrict its attention to the more loosely-coupled systems of classes (3) and (4) ignoring altogether those distributed computing systems involving parallel processors and multiple processors sharing a common memory.

The purpose of the ESMD security task is really two-fold. First, a capability for the simulation of different SYSCON architectures distinguished on the basis of their security properties will be demonstrated on the ESM loops. A communications security model supported at each loop node by CIE firmware was developed with this purpose inmind. The model allows for considerable freedom in the design of host-level policies for controlling access to communications channels and hence for controlling access to other network resources as well. Nevertheless, the range of SYSCON architectures that can be studied using this model is surely a very small subset of the distributed computer architectures that might be suitable for the system control application. The second goal, then, is to take a look at different SYSCON architectures on paper, considering their strengths and weaknesses in terms of cost, complexity, overhead due to security, approaches to access control, etc. The first problem of course, in making such a comparison is one of determining just what differentiates one SYSCON architecture from another. The dimensions of a SYSCON architecture which impact its security will be examined, and candidate architectures compared on the basis of this analysis. This aspect of the study will also suggest further simulation studies, impossible to conduct within the constraints of the communications security model discussed above, which should prove of value in any thorough comparison of SYSCON architectures.

There is no longer any doubt that network security encompasses issues of much greater scope than that of communications security alone. This report should make that much clearer. Nevertheless, the thrust of the ESMD security task is, in fact, directed towards the study of communications security because COMSEC is viewed as the very foundation for higher level network security policies. It is just as important to emphasize at the outset that the communications security model described in section 3.5 of the report is intended for use in areas secure from outside interference. Where procedural controls can protect communications lines against spoofing, traffic analysis, and other line tapping operations. Encryption obviously has a secure place in COMSEC, but if its overhead can be avoided and communication security maintained, so much the better. Where procedural controls are inadequate or nonexistent, encryption of course becomes necessary. An approach to network security which combines the two techniques is discussed in section 3.6 of the report.

The remainder of the report is divided into five major sections:

(1) What is security? Compares and contrasts the use of such terms as security, reliability, integrity, protection, and guaranteed service, stressing as well that security must always be viewed as a function of network requirements,

(2) Network Security: requirements and problems - discusses security requirements, problems, and approaches in the context of both single computer systems and computer networks,

(3) Dimensions of a SYSCON architecture lists those aspects of a network architecture which have an impact on its security, and so serves as an introduction to (4) and (5) below,

(4) The communications security model describes a simulation model, designed for possible implementation on ESM's CIE loop interface, which is to be used in network architecture studies related to COMMNET access control, and

(5) Security characteristics of alternative network architectures expands somewhat on (3) by considering in more detail such security-related aspects of network design as physical topology of the COMMNET, secure host processors, COMMNET protocols, and crypto controls.

3.2 What is Security?

The terms security, protection, integrity, and reliability are often used interchangeably, and to no purpose but confusion and misunderstanding. An informal description of these terms should help delimit the scope of the ESMD security task.

The principle of policy/mechanism separation (Lev 75) gives rise to the distinction between security and protection. To paraphrase (CS 76), a security policy is a set of rules which specify the access rights of subjects (processes and users, for example) to objects (devices and segments, for example) and/or information in a computer system. A protection mechaism is the means by which security policies can be implemented and enforced. For example, the protection mechanism of our communications security model, implemented in CIE firmware, guarantees that only authorized processes are allowed to modify the nodal conversion tables of the ESM loops; a host-level security policy determines just who those authorized processes are. The protection mechanism of the CIE prevents a host from transmitting messages to a given process unless it has the right to do so; such rights are granted to hosts only as a matter of host-level security policies. Though, as stated in (Lev 75), it is not always possible or desirable to separate policy from mechanism, for the purposes of this report the distinction between security and protection should prove a useful one.

To paraphrase (CS 76) once again, an integrity policy is a set of rules which specify the ways in which information in a computer system can be modified. Even if the information entered into a data base cannot be checked for correctness, it can be checked for plausibility (typically, a range of acceptable values for a given item would constitute an integrity check against incorrect entries). Just as for security, appropriate mechanisms must be designed to support whatever integrity policy is selected. Integrity violations occur in basically one of two ways. Unauthorized modification of information resulting from a breach in security generally results in an integrity violation: rarely in practice is information modified correctly under these circumstances. An integrity violation may also occur even when access to information is authorized - the information can be modified incorrectly either maliciously or by accident. The emphasis of this report is of course on security, and not integrity.

"Reliability is the probability that a system will perform satisfactorily for at least a given period of time when used under stated conditions: which failures occur, where, "failure" means unsatisfactory performance". (Alv 64) The distinction between security and reliability is seen to be a straightforward one. Not so obvious is the distinction made between reliability and correctness. Denning (Denp 76) states that "The distinction can be put this way: a system is correct as soon as it meets its specifications; it is reliable if it performs to satisfaction with high reliability. If the term "error" signifies a deviation from specifications, reliability means not freedom from errors but error tolerance."

"A system need not be correct to be reliable. It is considered reliable if the most probable errors do not make it unusable, and if the errors that do are rare and not at times of great need. Similarly, a correct system may be unreliable. Most correctness proofs make important implicit assumptions that can easily be invalidated in practice - e.g., that the underlying support system (such as the hardware) works correctly, that all data is consistent and correct, and that nothing outside a given subsystem can affect its behavior except via an interface."

The ESMD security task is not at all concerned with the general issue of network reliability, but it is concerned with the reliability of network security, and particularly with the reliability of communications resource management. Questions related to correctness of network security will be considered within the context of security assurance and proposed techniques for guaranteeing system correctness.

A common, though somewhat elusive, security requirement known as the guaranteed service principle (see Section 2.3 of the report) is closely related to the concept of reliability. The principle states that there shall be no unauthorized denial of service, i.e., "Users should never be denied access to objects (i.e., use of resources) to which they are entitled." (Neu 75). Where reliability, in its most general sense, is a measure of satisfactory performance, or service, the guaranteed service principle is a service requirement of a very absolute nature. A system may be reliable and still (probably unwittingly) deny its reliable service to some of its authorized users. Denial of service concerns as an aspect of network security will be discussed extensively in the report.

Though, as mentioned above, a security policy can be defined as a set of rules which govern the control of access by subjects to objects/information in a computer system, what is or is not viewed as security, in terms of the objects to be protected and the permissible modes of access accorded to subjects, is always a matter of network security requirements. One system can be considered more or less secure than another only if the security requirements to be met in each case are comparable. From this point of view alone is a security study of SYSCON architectures meaningful. A number of possible security requirements for system control will be considered in the report, and it is essentially these requirements that will form the basis for the comparison studies of the ESMD security task.

Finally, the study of SYSCON architectures and their security properties can only be one of several criteria to be considered in the selection of a network design for system control. A secure SYSCON architecture would hardly prove acceptable if it failed to meet reliability, throughput, and/or resource sharing requirements. Though not its focus of study, the ESMD security task will try to identify the flaws in a candidate SYSCON architecture which could restrict its usefulness as a basis for network organization, regardless of any desirable security characteristics it may have.

The current section has addressed the question of what is security in a very general way. Section three goes on to discuss the security problems and requirements of both centralized computing systems and general computer networks, first to suggest a dividing line between computer and network security, and then to motivate the choice of SYSCON dimensions which have an impact on network security requirements.

3.3 Network Security: Requirements and Problems

All security policies/protection mechanisms, whether applied to the security needs of a single computer system or to those of a network of computer systems, should make provision for

(1) Controlled access to resources
(2) Security assurance/monitoring, and
(3) Guaranteed service to users

A requirement for controlled access to resources encompasses not one, but several, important "security" requirements. The protection mechanism supporting a given security policy must be complete in the sense that every attempt to access a resource must be validated (at least indirectly) by the protection mechanism. Furthermore, the protection mechanism must be tamperproof, so that the protection mechanism is itself immune from unauthorized alteration.

Access control also depends on proper identification/ authentication of "subjects" (users, processes, devices, etc.) desiring access to the resources of a computer, or computer network. The term identification is used to mean the process of determining who or what an entity claims to be, and the term authentication the process of verifying this claim (Col 75). Once the identity of a subject has been authenticated (by using a password, for example), authorization checking on the part of the protection mechanism determines whether or not system resources may be accessed in ways requested by the subject.

Security assurance/monitoring is concerned with "providing assurance that (A) desired level of protection is maintained." (Col 75) Security assurance requires that a protection mechanism be certified correct and complete and that a security policy be shown to support system security requirements; monitoring the activity of a system throughout its lifetime provides an additional check on the integrity and adequacy of a security policy/ protection mechanism while the system is in full operation.

Finally, the "guaranteed service principle" holds that there can be no denial of service to authorized users. It appears that many cases of guaranteed service can be handled formally in much the same way as for access control (Neu 75), so that security assurance should be applied to denial of service concerns as well.

Before turning to the subject of network security, we will first take a brief look at security in the context of a single computer system. There is no attempt at completeness here, but we do hope to show where approachs to security in a single computer system can be applied to a network of such systems, and where the security requirements of networks depart from those of more tightly coupled systems.

Neumann et al (Neu 75) identify four principles of system security:

P1: There can be no unauthorized alteration of information (the alteration principle)

P2: There can be no unauthorized acquisition of information (the detection principle)

P3: There can be no unauthorized denial of service (the guaranteed service principle)

P4: There can be no unuathorized leakage of information (the confinement principle)

Of immediate interest here are principles P1 and P2. There are two commonly accepted approaches to access control in a computing system. Each provides mechanisms for complete user isolation and for controlled sharing of information among users. In one approach access control lists associated with each object in the system specify the modes of access allowed to the object by authorized subjects; an access controller, one perhaps for each type of object to be protected, mediates all attempts to access the object, and so enforces the authorization constraints of the access control lists. Access control list systems, like Multics, are generally implemented as a combination of both hardware and software, with access control lists interpreted in software, and tables of descriptors, which reflect the access constraints of the control lists, interpreted in hardware. The use of descriptors allows for secure access control without incurring the extra overhead that would result if every attempt to access an object were to be validated explicitly by an access controller.

Capability-based systems adopt a ticket-oriented approach to information protection. A capability is used to name, or identify, an object and to specify the rights of access to the object accorded the owner of the capability. "A capability is protected in that it is created by the system and cannot be modified or forged...when a capability is presented, it is interpreted by an accessing mechanism appropriate to the object referred to by the capability, namely, the type of object. Protection results from the enforcement of the rule that access to an object is permitted only upon presentation of an appropriate capability corresponding to that object...(thus,) authorization has meaning only with respect to a mapping from capabilities to the information in the system. Then the authorization to access a particular piece of information implies having access to an appropriate capability (itself obtained in an authorized way.)" (Neu 75) Despite the simplicity, flexibility, and efficiency of capability systems, they do have several potential problems largely avoided by access control list systems; in particular, how to control the propagation of capabilites among subjects, and how to rescind capabilities for an object regardless of their location in the system.

No matter what approach to access control is taken, the identification/authentication problem described earlier remains critical to secure access control. As stated in (Neu 75), "A user's authorization...depends largely on what is made available initially to the user at login. The identification and initial authorization are thus critical to security, but are beyond the scope of the operating system design - apart from the fact that this initial authorization can itself be made secure as a part of the operating system." The identification problem is especially acute in the area of communication security, and will be discussed more extensively in section 3.5 of the report.

Another aspect of secure access control corresponds to an extension of the authorization problem. In (Schr 75) Schroeder describes a scheme for passing capabilities from one protection domain (a generalization of the Multic's ring concept) of a process, via a procedure call, to another protection domain in the same process. The problem, of course, is one of ensuring that capabilities passed as arguments in the procedure call "be constrained to give no more access than is available to the calling domain." And the problem must be solved for the case of arbitrarily nested procedure calls as well. It will be seen that networks have their own special version of this n'th party authorization problem.

The confinement principle, security principle P4, adds a new dimension to access control. Confinement encompasses three potential security problems:

1. Leakage of information over normal channels. We have already discussed an instance of this problem in the context of capability-based protection systems. If a subject S1 allows another subject S2 to operate on its objects by passing appropriate capabilities to S2, then S2 cannot be allowed to retain the objects itself or transmit the objects to another subject unless permitted to do so by S1. (1) is actually implied by security principles P1 and P2.

(2) Leakage of information over subtle channels. As an example of (2), (Rot 74) discusses the possibility of transmitting encoded information over an "Invoice Channel". Consider the case of a proprietary service, available to the users of a computer utility, which uses a communications channel for the preparation of invoices. Billing information must be made available to both the lessor and lessee of the proprietary service; in order to ensure that the lessee receives the same copy of the bill as does the lessor (so that the lessee can observe the presence of any extraneous information and so detect potential security violations), all bills are sent by the service to a "Proprietary Services Administration," which then delivers identical copies of the bill to lessor and lessee. Thus, after performing a service for one of its customers, or lessees, the proprietary service can use the billing channel to transmit information about the transaction to the lessor of the service.

(3) Leakage of information over covert channels. It is also possible to access information illegally by making inferences from system behavior; for example, one subject can modify the working set size of a process so that another subject with access to system state information can interpret changes in working set size as a low bandwidth data channel. To paraphrase (Neu 75), although some cases of (3) can be covered by formal assertions for purposes of security assurance, the goal of eliminating covert channels is unattainable in a theoretically complete sense.

Confinement is a particularly insidious protection problem, and one that cannot be ignored in any network that claims to be secure. A network operating system, like the operating system of a single computer, must keep hidden as much as possible any system state information (on network resource usage, for example) that could be used to modulate a covert data channel. As important as the confinement problem is not network security, its solution is beyond the scope of this report - the simpler problems of access control are difficult enough.

This report has so far defined access control in terms of restricting access to objects such as devices, files, core memory segments, etc. The emergence of sophisticated data base management systems presents an alternative approach, or outlook, to information access control. Access control in a data base management system is not defined in terms of access to objects, and thus indirectly to their information content, but rather it is based directly on the information contained in a data base, without concern for the objects that may contain them. From another point of view, the individual relations that are part of every data base do in fact constitute separate "files" of information, and it is to these objects, the relations of the data base, that access must be controlled. As an example of security controls that might be desirable in a data base management system, the following is taken from (Dat 76). (Note: The terms file or record type, record, and record field may be substituted for relation, tuple, and domain, respectively, in the text below without any loss of understanding.)

Suppose that the database...includes a relation employee, defined on domains emp # (employee number), name, address, dept. # (department number), salarydate (date of last salary increase), and assessment (manager's evaluation of employee's performance). Then each of the following statements defines a reasonable level of access to this relation which should be granted for some particular category of user.

1. He has unconstrained access to the entire relation for all types of operation.

2. He has no access to any part of the relation for any type of operation.

3. He may see any part of the relation, but he may not change its contents.

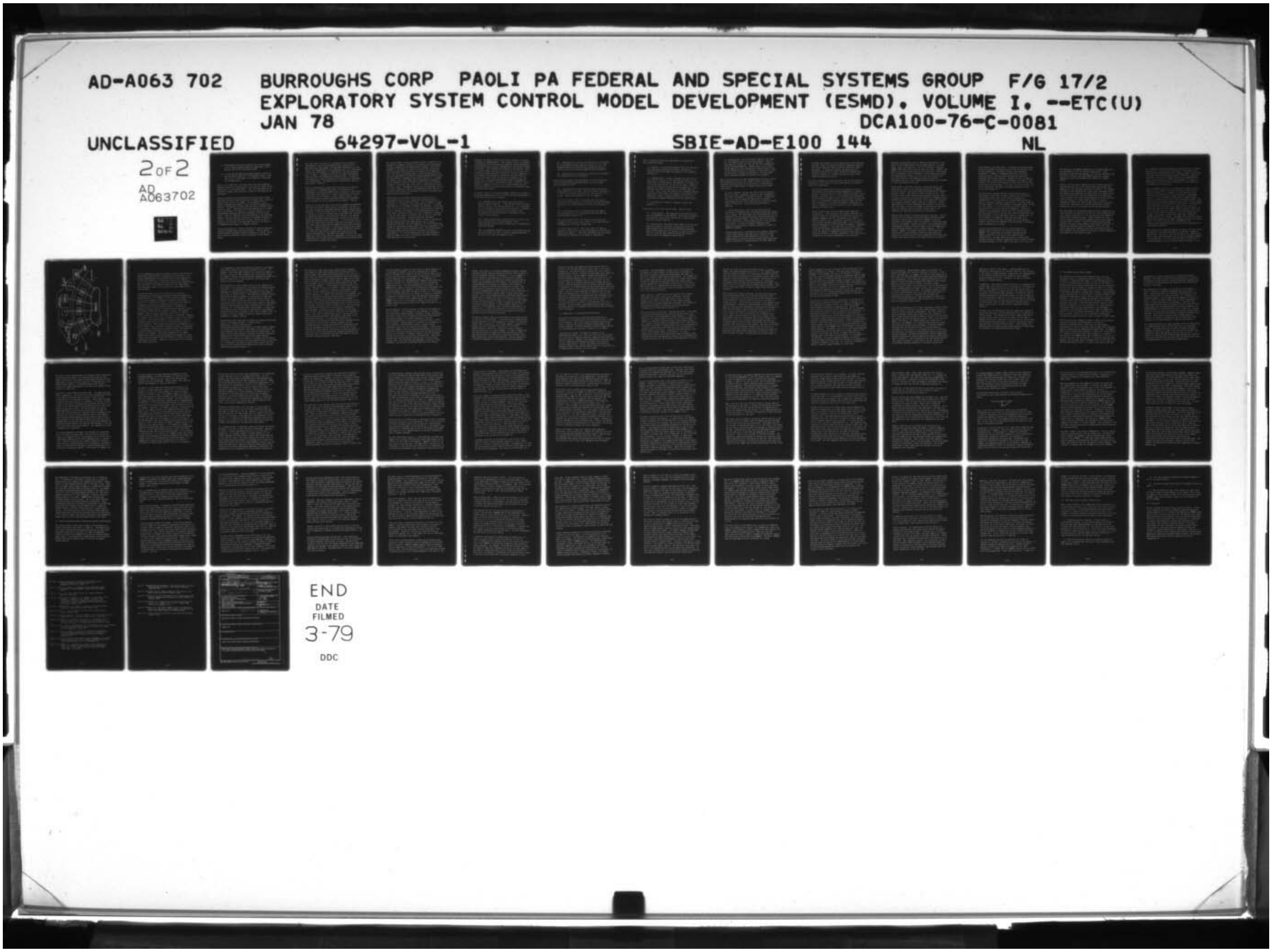
4. He may see exactly one tuple in the relation (his own), but he may not change it.

5. He may see exactly one tuple in the relation (his own), and alter some but not all of the values therein.

6. He may see the emp #, name, address, and dept. # domains, and within any one tuple he may alter only the name and address values.

7. He may see the emp # and salary domains, and within any tuple he may alter the salary value, but only between the hours of 9 a.m. and 5 p.m. and only from a terminal located in his payroll office.

8. He may see the emp # and salary domains, and within any one tuple he may alter the salary value, if and only if the current salary value is less than 5000 dollars.



AD-A063 702 BURROUGHS CORP PAOLI PA FEDERAL AND SPECIAL SYSTEMS GROUP F/G 17/2
EXPLORATORY SYSTEM CONTROL MODEL DEVELOPMENT (ESMD). VOLUME I. --ETC(U)
JAN 78 DCA100-76-C-0081
UNCLASSIFIED 64297-VOL-1 SBIE-AD-E100 144 NL

2 OF 2
AD
A063702

END
DATE
FILMED
3-79
DDC

9. He may apply statistical operators to the salary domain (e.g. to obtain average salary per department), but he may not see or alter individual values.

10. He may see the emp# and assessment domains, and within any one tuple he may alter the assessment value, if and only if he is the manager of the department identified by the dept# value.

There is still much work to be done before data base management systems can be built to handle such a wide range of access control constraints in an efficient manner. Just what mechanisms will be required to support these systems is largely an open question at this point.

Security principle P3, the guaranteed service principle, holds that there can be no denial of service to system users. Denial of service could result, for example, if the scheduler of an operating system, maliciously or not, prevented a program from running. "Trojan Horse" attacks often constitute attempts to deny service to users. "These may involve the implantation of clandestine side effects in a compiler, in a system routine, or in a user-produced subsystem that gains acceptance and use by other users." (Neu 75) Auditing of critical portions of the operating system is the only possible solution to the problem of denial of service. And even this may not be enough. Denial of service concerns will be shown to be especially crucial at the level of the communications subnet of a computer network.

Security principles P1 - P4 form the basis for security assurance in SRI's "Provably Secure Operating System." (Neu 75) Before describing SRI's approach to security assurance in detail, let us first review the problem of security assurance in more general terms.

The first step, of course, is to formulate a set of security requirements - security requirements being a function of a particular user environment. An important consideration in this step is the completeness of the requirement set, i.e., "The extent to which the requirements, if complied with, provide assurance that a system ... (designed on the basis of these requirements) does not contain any penetration vulnerabilities. It would be very beneficial to have techniques for establishing a requirement set's completeness. Such means do not currently exist and may never exist." (Sha 76)

Security assurance next requires the formalization of security requirements. "A mathematical model is a prerequisite to formulating a convincing argument that any system is truly secure. Such a model must demonstrate a sense of consistency that can be translated into the design of a secure computer system. This connection between the model and design provides the groundwork for an argument for the correctness of the system." (Sha 76)

Almost all of the work done in the area of mathematical modelling has been restricted to access control requirements, and even in this case the general confinement problem has been ignored almost entirely. In the Mitre Model (BL 73), for example, access control is defined in terms of an access matrix which specifies the modes of access that subjects are permitted to objects in the system. The model describes an abstract machine in terms of its machine states and allowable state transitions. It first defines what is meant by a secure state, and then constrains the machine to take only those actions which move the machine from one secure state to another. "Local Conditions" for authorized access strictly apply to a given subject and object, while "Global Conditions" specify the conditions under which a subject may access two or more objects simultaneously. The model thus makes provisions for supporting a multi-level security requirement in its use of global conditions and its association of classifications and need-to-know categories with subjects and objects.

The "Lattice Model of Secure Information Flow" of (Dend 76) is "A mathematical framework suitable for formulating the requirements of secure information flow among security classes. The central component of the model is a lattice structure derived from the security classes and justified by the semantics of information flow. The lattice properties permit concise formulations of the security requirements of different existing systems and facilitate the construction of mechanisms that enforce security." In (Dend 76) Denning states that "most control mechanisms are designed to control immediate access to objects without taking into account information flow paths implied by a given, outstanding collection of access rights. Contemporary access control mechanisms, such as are found in Multics or Hydra, have demonstrated their abilities to enforce the isolation of processes for secure information flow among cooperating processes."

Denning goes on to say that "the primary difficulty with guaranteeing security lies in detecting (and monitoring) all (information) flow causing operations. This is because all such operations in a program are not explicitly specified - or indeed even executed; as an example, consider the statement: If A=0 then B:=0; if B is not equal to 0 initially, testing B=0 on termination of this statement is tantamount to knowing whether A=0 or not. In other words, information flows from A to B regardless of whether or not the then clause is executed," and B's security class must be updated accordingly. Denning's model was formulated specifically to account for such "implicit" flows of information, and she contends that "by decoupling the right to access information from the right to disseminate it, the flow model goes beyond the access matrix model (used in the Mitre approach) in its ability to specify secure information flow. A practical system needs both access and flow control to satisfy all security requirements." It will be interesting to see what application Denning's model will have to the general confinement problem described earlier in this section.

Neumann et al (Neu 75) address the issue of how best to proceed from model to implementation. Their primary intent is to suggest a methodology which facilitates all phases of system development: system specification, design, implementation, and proof of system correctness. They recognize the enormity of the problem. "There is a general skepticism about the feasibility of ever proving a large-scale operating system - because of the sheer complexity of the programs, the problems of concurrency, and difficulty of stating assertions on what the system is intended to do. The skepticism is certainly justified with regard to contemporary systems such as OS/370 or the present version of Multics...nevertheless, it appears realistic to us to prove properties of an operating system that is designed according to a methodology that facilitates such proofs."

As stated in (Neu 75), system development involves five stages of design and implementation (S1 to S5) and five associated stages of verification (V1 to V5), as follows:

(S1) Decomposition of the system into a hierarchy of abstract machines (M0, M1, ... MN, with M0 the most primitive machine, and for each abstract machine MI, a set of programs PI can be thought of as interpretively executed to implement MI+1), selection of functions for each machine, and determination of which functions are available at which levels in the hierarchy.

(S2) Formal specification of each function in terms of the value returned and/or the state changes. These specifications take the form of assertions in a nonprocedural assertion language.

(S3) Correspondence between the state of each machine MI and the state of MI-1 for I>0 (in terms of "mapping functions" written in the assertion language of S2)...

(S4) Implementation of each of the functions of each machine MI as a program using the functions of MI-1 for I > 0. The implementations are called abstract programs (since their execution need not be directly supported by any hardware).

(S5) Implementation of all primitive functions as programs in the instruction set of the hardware...

The verification phase of the methodology is closely integrated with the design-and-implementation phase, with five stages associated with S1 to S5.

(V1) Establishment of global assertions about the desired system behavior to be proven. One use of global assertions is to define the necessary constraints for the security of the system. (Security principles P1 - P4, for example)...

(V2) Verification of S2: Verification that the specifications are self-consistent at each level, and that the global assertions of V1 follow logically from the specifications of S2...

(V3) Verification of S3: Verification that the mapping functions defined in S3 are consistent with each other and with the specifications in S2.

(V4) Verification of S4: Verification that the abstract programs in S4 are correct with respect to the specifications and mappings of S2 and S3, respectively...

(V5) Verification of S5: Verification that the abstract programs in stage S4 are correctly implemented in the hardware instructions (Stage 5). This stage guarantees that the system assertions are satisfied by the system as implemented.

(Neu 75) describes three basic approaches to the design of a secure operating system:

(1) Patching - use an existing system, detect its deficiencies (possibly by penetration studies), and patch it. (Various studies have shown this approach to be completely ineffective.)

(2) Language Design - design a language that can intrinsically enforce security via its own restrictiveness, or via compilation or interpretation, and implement a translator for it in a way that enforces its use for all users. (Dend 75) suggests two possible limitations to this approach. "First, (information) flows not specified by a program cannot be verified. Such a flow could result from a language implementation defect that allows, for example, array bounds to go unchecked...second, a program certified as secure can be transformed by hardware malfunction into an insecure one.")

(3) System Design - design or redesign a system, and implement it.

System design itself takes several forms. (Neu 75) lists:

(1) Virtual Machine - The system kernel provides each user with an independent virtual machine. The primary drawback to this approach is due to the restrictions it places on the sharing of system resources.

(2) Kernel Design - The system kernel handles only the most primitive protection functions, leaving the rest of the system unspecified. Note that the methodology described above could be applied to the design of a "Security Kernel." (Neu 75) cites the Hydra operating system and the Mitre security kernel as examples of kernel design.

(3) System Design - Unlike the kernel approach, this calls for the complete design of an operating system. (Neu 75) cites Multics, Plessey 250, CAP, and BCC 250 as examples of the system design approach. Neumann opted for the system design approach as well, arguing that "Specification of all visible system functions is necessary to make proofs of user-oriented security properties meaningful. For this reason, Kernel approaches and partial designs were eschewed."

Once the system design has been implemented and verified, surveillance mechanisms are required in order to detect malicious or accidental attacks against the system while it is operational. (Sha 76) gives a good account of computer system surveillance requirements:

> In current computer systems, there are protection measures which are either incompletely defined or made incomplete as the result of hardware or software failures. Consequently, there is a need for additional deterrents to attacks... surveillance provides such a deterrent and involves two major types of activities: threat monitoring and security auditing.
>
> ...(Threat monitoring is the real-time monitoring and detection) of attempted and/or successful penetration attacks on the system. The requirements for these mechanisms should specify the set of events that should be monitored during the system's operational phase. The threat monitoring activity involves notifying a security administrator of the existence of an attack. It must be conducted immediately upon detecting an attack and in a manner sufficient to ensure an immediate response.
>
> (Security auditing) is concerned with the logging, reduction, ex post facto analysis and reporting of security related events indicative of system penetration activities...the logging activity involves collecting system operational data which is indicative of system attacks, but which cannot be

> analyzed and responded to until sometime after the attack has occurred. Thus, the amount and type of information required for security auditing differs from that needed for threat monitoring. It typically involves information which is indirectly indicative of attacks and must be obtained from extensive analysis of a system's operational states, data inputs and processing results.

As a final parameter of system security, (Sha 76) also notes the concern for the reliability of the access control, or internal protection, mechanisms:

> Failure of a system occurs due to the presence of errors in both hardware and operating system components of a computer system. A prerequisite for a secure system, therefore, is the absence of software errors (in the access control mechanisms) and a minimum of hardware errors. At this point, we note that there is a fundamental difference between hardware and software reliability in the sense that hardware components can never be completely error-free because of their physical and operating characteristics.
>
> Errors can be divied into two classes, algorithmic and probabilistic. (Both) hardware and software components of a computer system are (susceptible to algorithmic errors, but) hardware components are also susceptible to failures of physical elements (e.g. transistors, registers, etc.) which occur during a system's operational phase. Such failures are usually accounted for by the following techniques: redundancy (static and dynamic), fault detection and location, fault tolerance, and dynamic reconfiguration/recovery. Algorithmic errors are design or implementation defects which occur in either the hardware or software components. Such failures are accounted for by software redundancy or certification.

Reliability requirements were an important consideration in the design of the communications security model applied to the ESM loops. This is particularly true of the "multiple controller" COMMSEC model described in section 3.5of the report. However, the model is not so much concerned with the reliability of the access control mechanism, which is embedded in CIE firmware and protects the communications resource, as it is with the reliability of hosts which establish the communications security policy determining the way in which communications channels are allocated to hosts and devices on the loop.

Access control, fairness of service, security assurance, and reliability - these are the important parameters of computer security, both for centralized computing systems and for computer networks as well. The security issues discussed so far in the context of centralized computing systems become important issues of network security not only because networks are composed of such systems, but also because these same issues emerge when networks are treated as complex computer systems in themselves. Network requirements for resource management and protection can be very much like those of centralized computing systems; the distributed nature of computer networks, however, greatly adds to the complexity of supporting these security requirements.

Several recent studies illustrate this point very well. (Cos 75) describes an experimental distributed network operating system, called the resource sharing executive (RSEXEC), which was developed in an attempt to pool the resources of individual Tenex hosts on the ARPANET. Resource sharing is thus accomplished over a homogeneous network of identical hosts, with each supporting a Tenex operating system on top of a PDP-10 computer. (Cos 75) states: "RSEXEC is designed to provide an environment which allows users to access network resources without requiring attention to network details such as communications protocols and without even requiring users to be aware that they are dealing with a network...for many users, ...it should not actually matter which host provides the Tenex service so long as the users could do their computing in the manner to which they had become accustomed. A number of advantages would result from such resource sharing. The user would see Tenex as a much more accessible and reliable resource. Because he would no longer be dependent upon a single host for his computing, he would be able to access the Tenex virtual machine even when one or more of the Tenex hosts were unavailable. Of course, for him to be able to do so in a useful way, the Tenex file system would have to span across host boundaries."

This observation suggests one way in which network security requirements, though much the same in general as those of more centralized systems, give rise to greater complexity of implementation. A network-wide file system should provide the same file protection features usually found in the file system of a single computer. The basic problem, as explained above, is that the file system, i.e., the file access controller, must be distributed over an entire network of computers. Information related to the protection state of each file (indicating who or what may access the file, and how) may itself be distributed, and the consistency of this information, even as it extends over an entire network, is essential. What would be viewed as an integrity problem in the case of normal user files becomes a security issue when the consistency and correctness of protection state information are at stake.

(Cos 75) also describes the use of a distributed, multi-computer access controller for controlling access to the ARPANET. The intent here is to achieve load sharing and higher reliability in the area of user identification/ authentication. For many users the network is usually made available through a terminal concentrator device called a terminal interface processor, or TIP. When a user logs on to a TIP, the

TIP connects to the RSEXEC subsystem of an available Tenex host, and the RSEXEC will act as a network logon server. After supplying a valid name and password, the user can continue to use the TIP, the network, and other RSEXEC facilities. Network complexity in this case enhances the reliability of access control.

Finally, (COS 75) states: "Experience with the ARPA network has indicated the need for access controls above and beyond those supported by the constituent host service machines. For example, an access control mechanism has recently been implemented within the communication subnetwork to allow the set of a network hosts with which a particular host can communicate to be administratively limited. The access controls applied to the TIP also fall into this category. In many cases the goals of network transparency and ease of access conflict with those of security and privacy. Each security or access check places a barrier between the user (or his program) and the desired resource."

The requirement for network-wide access controls is precisely the issue that most concerns Cole in (Col 75). Cole proposes the use of a "security controller", or SC, through which all requests to access remote objects must pass. The SC maintains an access control table which specifies the modes of access that subjects are permitted to objects. Typically, subjects would include persons, terminals, and processors; objects would include all or some of the subjects, plus host computers, files, etc. The initial design only concerns itself with controlling access to host computers, but the design, Cole claims, is easily extended to the protection of the objects as well.

In a completely centralized approach to access control, a network would contain a single security controller. The access control function can be distributed across a network, however, by dividing the network into non-overlapping security "domains", consisting primarily of hosts and terminals connected to a communications subnet, with one security controller per domain. For ease of description we will confine our discussion of Cole's approach to intra-domain interactions only.

Figure 3-2adapted from (Col 75), illustrates a single security domain within a network. Terminals and hosts, as well as the security controller for the domain, are all connected to the communications subnet via so-called intelligent cryptographic devices, or ICD's. Meaningful communication between network entities is possible only if their ICD's have matching keys. The ICD's contain a key select mechanism which selects either a common network-wide key, a private key, or a working key. Initially, the ICD of the security controller, or master ICD, is set to the common key, while the other crypto devices, the slave ICD's, are set to their private keys. When a host or terminal reuires the services of the security controller, its slave ICD must first transmit to the master ICD over the common key a "HELLO/ID" message identifying the slave. The master ICD is the only crypto device which recognizes the HELLO/ID message and knows the private keys of the other crypto devices. Consequently, all slave-to-slave communications paths are initially blocked by the crypto device themselves.

Upon receipt of a "HELLO/ID" message, the master ICD looks up the private key of the slave and uses it to send back a temporary working key. This key is then used to encrypt the access request sent to the security controller by the host or terminal connected to the slave ICD. It is the job of the security controller to determine whether or not the request should be honored. If so,

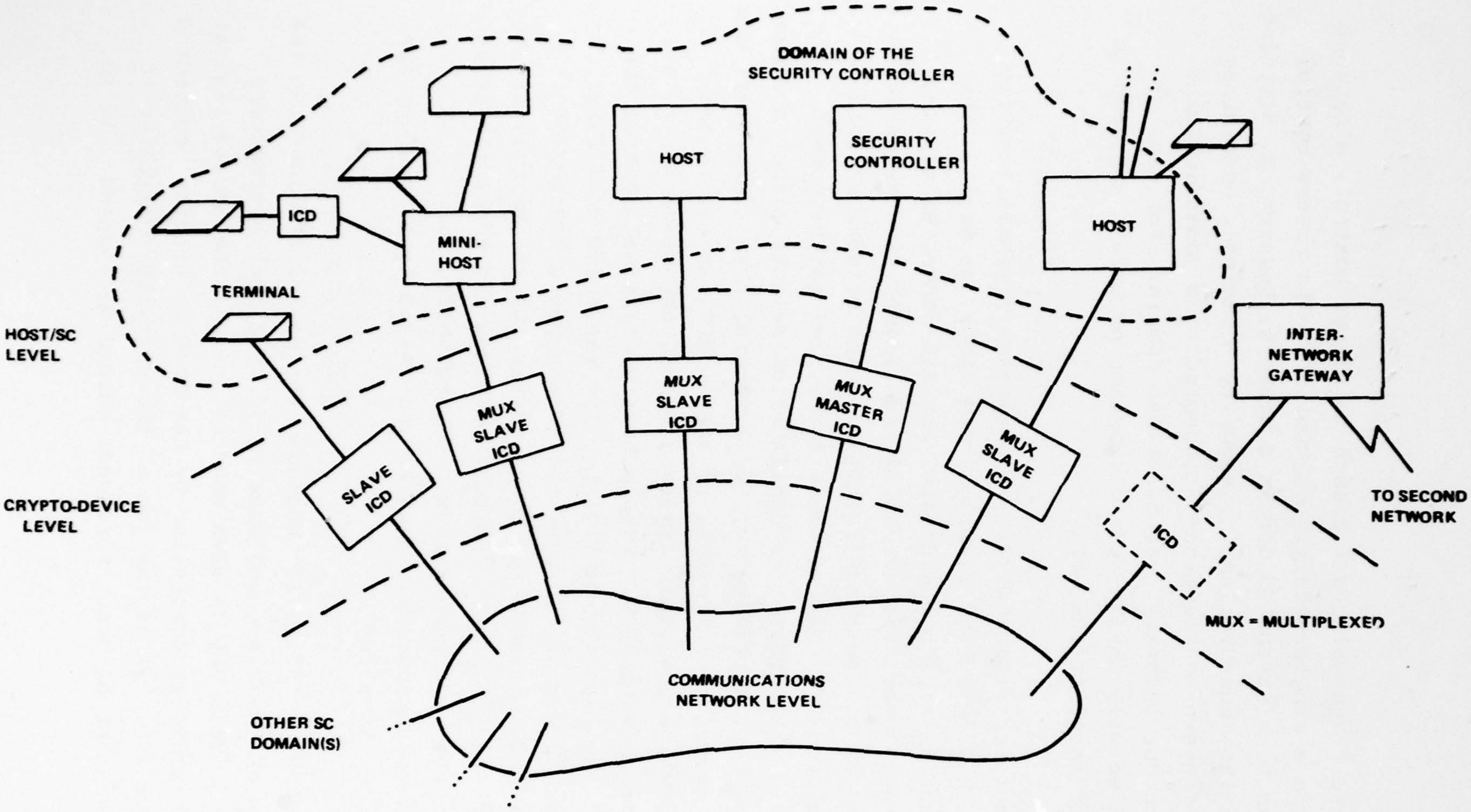


Figure 3-2. The Levels Involved in a Secure Network

the SC establishes a connection between requestor and resource by distributing working keys to the appropriate slave ICD's. All network communication, then, proceeds only with the approval of the security controller. In this way, the security controller can, if necessary, prevent two network entities from communicating with each other - even though they belong to the same network. This protection feature is also found in our own communications security model.

As described above, the security controller is primarily responsible for establishing authorized connections between two network entities. For example, a host A, acting on behalf of one of its users, requests a connection to another Host B whose services are required by the user. In this case, the requested resource is in fact host B, and access to the resource is permitted by the security contoller when it establishes a connection between A and B. The user will then interact directly with Host B, requesting any resources it owns. Cole suggests, however, that the security controller could also be used to control access to objects other than hosts or terminals. Suppose, for example, that a user at Host A needed to access a file located at Host B. Host A would transmit an access request to the SC, identifying the user, the file to be accessed, and the type of access requested. The SC would have sufficient information in its access control table to approve or deny the request as necessary. If the request is honored, the SC notifies Host B and establishes a working connection between the two hosts, permitting file transfer, for example. The primary advantage of the second approach over the first is that unauthorized requests for remote objects can be detected before any connection is made to the host containing the resource. Security is less likely to be violated under these conditions.

It is important to note that hosts participating in the network could easily subvert the protection measures of the SC. For example, a host could send requests for a resource on behalf of one user, but in the name of another user with greater access privileges. Clearly, network hosts must themselves be shown to be "secure" before there is any assurance of the network-wide security in Cole's approach.

The particulars of Cole's scheme are not of real importance to this report, but his approach to the analysis of network security problems is. He emphasizes throughout his report (Col 75) the importance of examining security requirements as they apply to every level of network design. This approach to network security constitutes an obvious extension to the problem of supporting security on a single computer. His design consists of three levels: The host/security controller level, the ICD level, and the communications subnet level; the general security requirements issues he discusses are the issues of access control, security assurance, etc., already described in this report. At each level of his design, Cole reconsiders the same security requirements, determining just what is needed at each level to support them. In so doing, he raises a number of security issues of particular importance to computer networks.

Authentication and authorization problems are especially difficult ones for the network designer. In Cole's words:
"Networking...creates identification/
authentication problems beyond those of a single computer system. In the multi-system (network) environment, the various systems (host computers) must also be identified and authenticated. One aspect of this issue is whether processes on the hosts should be considered as entities which require such identification/ authentication, either as a requester of network services and/or as a server. Since the host will have complete access to the data of its processes, including any authenticators which they might have, the use of process level authenticators does not protect against malicious host behavior..."

Cole goes on to raise other design issues as well regarding distributed vs. centralized authentication checking (his is "a hybrid scheme, with checking being distributed to the level of a given region or domain, but being centralized within each domain") and the problem of n'th party authentication ("when one site must operate on behalf of another, which itself is operating on behalf of yet another, but in all cases for some ancestoral user who initiated the request"). On the subject of authorization, Cole states: "Several entities are involved in alsont every computer transaction, e.g., a person, a terminal, a host computer, and a process. Each of these entities must be authorized to either receive, process, or transport the information being handled. The logical intersection of these authorizations will establish the level of information which can be sent via this sequence of entities, but a further step-by-step authorization check is also necessary to ensure that only the proper entity (or entities) are the ultimate recipients of the information, e.g., one entity may be authorized to process, but not to copy the information... In some instances, a requester will be connected to a host which will, in turn, need to access other resources on the requester's behalf. This need can interactively grow to the general n'th party authorization problem. Two different approaches (to the problem) are possible: (1) continually subsetting the authorizations as necessary so that the final privileges are the intersection of those of the original requester and all intermediate nodes, thereby ensuring that no intermediate node gets any information for which it is not authorized, and (2) handling the authorizations iteratively on a pair-wise basis, so that the n'th level will provide any requested information for which the n-1st is authorized, and leave the burden of further controls of passing of data to that host."

Cole also mentions a number of other security issues important to a networking environment, such as the distribution of access control information in a network, the use of network audit centers for security monitoring, failure modes of operation of the access control mechanism, error containment, etc. In addition, he discusses the security requirements most commonly associated with computer networks, i.e., those related to the use of physical communication lines. The most pressing security issues at the COMMNET level concern the design of authentication mechanisms, and protection against line tapping, traffic analysis, playback of recorded messages (spoofing), and denial of service. As discussed earlier, Cole uses encryption as a means of protecting information transmitted on communications lines, but encryption alone cannot guarantee the security of the communications net. The problem of COMMNET security will be discussed at length in sections five (COMMSEC model description) and six (the comparison study) of the security task.

Another approach to network security is based on the concept of a secure processor. As an example of this approach, we consider here briefly the network design of Fitzwater, Kramer, and Cowan as described in DCA's Unified Software Architecture Study (USAS 76). The design proposed in the study represents a much more distributed approach to access control than Cole's does. Unlike Cole's design, it imposes specific constraints on the behavior of host processors, and they must be shown to abide by these constraints before they can belong to the network. In particular, hosts must support protection mechanisms designed by the network architects to protect resources even as they are passing from one process in the network to another, whether or not the processes sharing the resources are located at the same host. (This does not mean that the same physical machine must be used at each network node - only that each node implement, by whatever means available, the protection mechanisms required by the network design. The result is a virtually homogeneous network of secure processors having identical protection features from the point of view of processes running in the network.) If, for example, a process chooses to allocate one of its resources to another process in the network, the allocating process may associate an access control program with the resource - the program will be invoked whenever the resource is accessed by the receiving process, no matter which host now contains the resource. The access control program may restrict access to the resource in arbitrarily complex ways; access control need not be restricted to the simplistic read/write/execute modes of more conventional systems. Since a process may associate a different access control program with a resource each time it allocates the resource to another process, the security policy of a process is thus defined in terms of the processes it allows to access the resource, and the access control programs it uses to mediate access to the resource. In this approach, then, protection comes down to the process level - a much finer granularity of protection than is provided in Cole's design, which can only guarantee host-level protection with its security controller.

In addition, process rights are also viewed in the network design as resources to be protected. If a process owns a communications resource allowing it to send messages to a process at a different network node, it may do so. If no such resource is owned by the process, the host containing the process will not allow communications to take place. There is no need to consult a security controller as in Cole's scheme. To repeat, though, each host must be proven to support the network design constraints which would ensure the protection of resources according to the security policies of host-level processes. And the assurance is an essential ingredient in the design of any secure processor.

Enough now has been said about the general network security problem to put the work described in this report in its proper prospective. We could not hope to address all of the issues discussed above, nor do we intend to discuss many other aspects of network security, such as tempest testing, administrative controls, physical security, etc., which have little to do with the protection of information once it is introduced into the network. What we do intend to discuss will be made clearer in section 3.4 which identifies some of the more important aspects, or dimensions, of a System Control network architecture which have an impact on security as it has been informally described in this report. Section 3.5 then presents our communications security model and indicates how it may best be used as a tool for the simulation of System Control architectures distinguished on the basis of the architectural dimensions outlined in section 3.4. The model, as explained in section 3.5is not by itself sufficiently general to allow all such dimensions to be fully explored. For this reason, other important dimensions will be discussed outside of the context of the security model in section 3.6 of the report.

3.4 Dimensions of a *System Control Architecture*

This section serves only to identify those dimensions, or parameters, of a System Control architecture which have an impact on its security. The list of security-related features given below is based on the survey of security requirements of the previous section. A fuller discussion of many of these features will be deferred until sections 3.5 and 3.6 of the report.

Interconnection scheme. The physical topology of the communications subnet (e.g., loop, star, bus, circuit switched, etc.) will have an impact on security, especially with respect to denial of service. Moreover, the interconnection protocol used to initiate communications between two network nodes may have security implications. Logical interconnection constraints imposed by higher level controls affect security in ways that will be demonstrated in section 3.5 of the report.

Protocols. Protocols/schemes (at every level of a network) for the purpose of identification, authentication, authorization, detection of spoofing, connection establishment and usage, etc., may differ radically among network architectures. For example, we have already seen how Cole uses encryption as a means of providing a "secure" communications channel between two network nodes - another approach will be described in section 3.5 of the report, when we present our communications security model. Farber's (FL 75) secure network protocol will also be discussed (in section 3.6) to illustrate this point.

Crypto Controls. Crypto controls are required in any network where physical security alone is not sufficient to prevent outsiders from tapping the communications lines. Our own communications security model does in fact assume adequate physical security against tampering by outsiders, but schemes for integrating the model with link and end-to-end encryption techniques will be described in section 3.6.

Resource Protection Facilities. Networks may differ in the kinds of resources guaranteed protection by the network, and in the operations that can be performed on these resources. A "secure" processor in a network supporting multi-level security (MLS), for example, is expected to protect devices, files, memory blocks, etc., against illegal access by unauthorized users. Each subject and object is assigned a classification level, so that subjects of a lesser classification level than that for a given object are denied access to the object. Authorized subjects may have read, write, and/or execute access to the object. In a more sophisticated system, users could define their own object types and associate with them appropriate access control procedures and resource operations for the protection and manipulation of these objects, as is the case with the network design of Fitzwater, Kramer, and Cowan discussed in section 3.3. For example, messages could be treated as objects to be protected by networks hosts so that only certain processes with the necessary access rights could read/write sensitive portions of a message. The protection of objects is a particularly difficult problem in a networking environment since the same protection guarantees must apply to an object even as it is moved from one network node to another. Each level of the network must be shown to support these protection guarantees.

Nature/location of security controls. We have already seen how some network designs make use of "secure" processors, whose security properties are well-defined, while other designs do not, relying instead on controls external to hosts for purposes of network security. Cole's security controller is an example of the second approach. Fitzwater's network design, following the first approach, specifies the constraints on host behavior necessary to make guarantees about resource protection down to the process level (see our description of his design in section 3.3). In Cole's scheme, the security controller can be depended on to protect hosts from one another (at least with respect to the establishment and use of authorized host-to-host communications channels), but Cole can only make vague assumptions about the trustworthiness of hosts in their interactions with the security controller and other hosts in the network. The basis for network security in this case depends largely on the verification of security-related host behavior without specifying the protection mechanisms and system design constraints, as Fitzwater does, to ease the burden of proof. Our communications security model is another example of the use of security controls external to the hosts and devices of a network. The loop access controls are embedded in CIE firmware, and not in the processors attached to the loop. This was necessary because no assumptions could be made about the behavior of devices attached to the loop - some devices would not have been intelligent enough to support the necessary controls, while other devices with the necessary intelligence (the PDP-11's, for example) may not provide the requisite architectural features for assuring the correct, tamperproof, and uncircumventable operation of the security controls. Embedding the controls in CIE firmware physically isolates devices from controls so that such guarantees regarding security controls on the loop can convincingly be made.

Distribution of Access Controls. The degree of distribution of access controls (whether they apply to files, to devices, or to the communications medium itself) is an important parameter of a secure network architecture, especially as it affects the distribution of the access control data base, and the throughput and reliability constraints on the access control mechanism. As described in section 3.3, Cole provides for some measure of distributed access control in his network design, but control of resources is centralized within any given domain of the network. Fitzwater's network design (see section 3.3)permits a much greater degree of distributed control than this. In our communications security model, each node acts as a loop access controller so that access control at the COMMNET level is distributed to the fullest extent possible. Furthermore, there are no constraints whatsoever on the distribution of access controls at higher levels of the network - this allows any variety of network architectures distinguished on the basis of this parameter to be simulated on the ESM loop. Like Cole's approach, however, no explicit security-related constraints have been placed on the operations of processors attached to the loop, i.e., we have not attempted to define a "secure" network processor in this study.

Error Containment. Measure must be taken in any network to minimize the impact of a security violation on the net. In the case of the ESM loop, for example, the CIE at a node might fail in such a way as to allow messages with illegal destination addresses to be transmitted. Such violations should be detected as soon as possible, before they have an undesirable impact on other network nodes or on higher levels of the net. Of course, though provisions must be made in the architecture for the containment of errors, schemes should be designed so that, even in failure modes, no security compromises can occur. For example, hardware failure of an LIU (Line Interface Unit) at a loop node should result in the electrical isolation of the node from the loop, thus eliminating any possiblility of spurious mesages entering the loop. Service may be denied to some network users, but network security otherwise remains intact.

Heterogeneity/homogeneity of the Net. The extent to which network nodes are alike or different strongly affects network security in terms of security assurance requirements. The task of security assurance becomes easiest when network hosts are identical, i.e., when a single physical system is used at each host site, with each host supporting the same operating system. This is the case with RSEXEC (see section 3.3), which consists of a collection of PDP-10's on the ARPANET, all running the Tenex operating system (and identical network control programs). Fitzwater's design certainly does not preclude the use of identical physical processors at host sites, but its primary intent is to allow for the possibility of managing a heterogeneous network of dissimilar physical machines. The design establishes a logical homogeneous network by defining an abstract machine and requiring each network node to emulate it. User processes see the same logical machine (and network interface) at each node in the net, so that the network-wide software portability is achieved; perhaps more important in terms of this report, a uniform scheme for the manipulation and protection of resources, wherever their location in the network, is made possible. Nonetheless, the implementation of the logical machine must be certified for each different physical machine in the net. Typically, the job is difficult enough for a single machine.

Transparency. The possible effect of security on transparency has already been noted in section 3.3 of the report in the context of RSEXEC. The transparency of the net vis-a-vis the user may, in turn, have an affect on network security. The results of network operations that should be made available to the end user must be carefully considered in any network design, or covert channels of the kind described in section 3.3 will be unearthed and exploited by the malicious user. The same care should be taken in the design of every network level so that information reflected back to higher levels could in no way be used to compromise network security.

The above list of architectural features, admittedly incomplete, gives some indication of the all-pervasive effect of security on network design. Our communications security model, described in the next section, was to serve as the basis for the study of SYSCON architectures, whose security characteristics varied along the dimensions described above. It has its limitations, however, and these will be carefully noted as well. Later sections will address issues of network security which cannot be treated within the context of the communications security model.

3.5 The Communications Security Model

The communications security model described here in no way represents a best, or necessarily complete, design of a secure interface to a communications subnet. It was designed primarily for use on the ESM loops as a simulation tool for the study of SYSCON network architectures having different security properites, and there has been no attempt to optimize the model for a given architecture. Nor does the model concern itself with the problem of encrypting information injected into a network. The assumption is that physical security is sufficient to prevent outsiders from tapping the communication lines. Security controls in the CIE interface to the loop, as defined by the model, guard against (certain kinds of) malicious behavior on the part of any devices attached to the loop. If several such "secure" loops (or, more generally, networks) are to be integrated into a single network, and physical security alone (due perhaps to the geographic isolation of the individual loops) can no longer be expected to protect the net from outside interference, then crypto controls should be used. Section 3.6 will investigate the problem of incorporating crypto controls into such a network of secure loops.

The previous section suggested a number of ways in which network architectures could be distinguished on the basis of security-related attributes. Obviously, the term network architecture is used in a much more general sense than to mean the physical topology of the communications subnet alone. No COMMSEC model could be sufficiently flexible to explore all the dimensions of a network architecture which have an impact on its security. A discussion of our COMMSEC model, what it does best or not at all, will be given later in this section. It should be mentioned here, however, that our model does not depend in any critical way on the ESM loop architecture itself. It could just as easily have been implemented on a network architecture of completely different topology. The extent to which the security properties of network architectures having different logical and physical topologies can be studied on the ESM loop will be discussed in this and the next section of the report.

The description of the model, as given below, will largely be given in terms of its possible implementation on ESM loops 1, 2 and 3. Minor differences in the possible implementation of the model on loop 4 will, of course, be noted. It is important to recognize, however, that the model was developed with loops 1-3 in mind. It had to be simple enough that it could be implemented without too much difficulty in CIE firmware. Obviously, more sophisticated security-related simulation features could have been incorporated into the powerful network interface provided by the "ALGOL Machines" of the BDS nodes in loop 4. Then, too, security controls in the CIE could not become too elaborate without making difficult the simulation of potentially interesting network architectures. Those security features were included which were thought to be useful in simulations, and not necessarily final solutions to the general problem of communications security.

Our communications security model provides a means by which access to a communications subset can be controlled in a distributed fashion. The model specifies the nature of security-related information required at a node's CIE, and the ways in which this information can be manipulated by the CIE (CIP in loop 4). In terms of the general security models described earlier, each CIE serves as a controller for the loop object by monitoring the flow of information both to and from its attached device.

The CIE of a given node maintains its security-related information (as defined by the model) in an access control table, or ACT. There is one entry in the table for each network logical ID (LID). Essentially, an LID serves as a name for a termination point of a communications channel. As described below, there exists a hierarchy of rights associated with the ability to modify the LID entries of a node's act.

Each LID at a given node has a single owner. Its owner has three basic capabilties. First, he may set the corresponding LID entry in the node's conversion table (in loop 4, he may set the appropriate bit in the process ID matrix). This allows the owner to specify the nodal location, or residence, or the LID. Second, he may set the source and destination sub-fields, each two bits in length, of the LID attribute field associated with the LID in the node's access control table. The two sub-fields are used to determine whether or not the LID can be used at the node as a legal local source address (the node may send out messages to the loop with this LID as a source address), remote source address (the node may receive messages from the loop with this LID as a source address), local destination address (the node may receive messages from the loop with this LID as a destination address), or remote destination address (the node may send messages to the loop with this LID as a destination address). Any combination of uses is possible for a given LID.

As a third capability, the owner of an LID may set the delivery bit of the LID's attribute field. The delivery bit allows the owner, on a temporary basis, to discontinue its use of the LID as a destination address in CIE control messages sent to a network node. Setting the delivery bit, however, explicitly indicates that a security violation should not be signalled at the node as a result of receiving from the owner a control message which uses the LID as a destination address. In this way, for example, the

LID can be used as the remote destination address of a CIE broadcast message, to be sent throughout the network, without causing error conditions to be raised at the few nodes not intended to act on the message. Network nodes, then, need not receive such messages on a node by node basis.

The owner of an LID, some host-level process running in the network, is itself identified (indirectly) in the ACT by the logical ID it uses as a source address for control messages related to the LID it owns. For example, suppose the CIE at a given node receives a control message from the loop, specifying that the residence of some LID be changed. The source address of the message, as explained above, indicates who the supposed owner of the LID is. If the source address matches the owner's LID as specified in the access control table, then the CIE will carry out the request. Otherwise, the request is denied. A process may use any one of a number of LIDs in its communications with other nodes, and it is only by this association with logical IDs that its capabilities are known to the CIE loop access controllers. Capabilities, then, are associated not with process identifiers per se, but rather with the channels (or, more precisely, the source addresses) they use in communications with the CIE's. The ability to use a given channel is implicit authentication of the identity of the process using it. This approach is thought to simplify CIE design and improve run-time efficiency considerably. Thus, it is up to higher level controls to ensure that these channels are used only by authorized processes. (In the absence of such controls - in a network, for example, whose hosts cannot prevent their processes from using any channels they choose, and channel control is only possible on a node level basis - the capabilities listed in the act would have to be viewed as belonging to a network node, and not to one of its processes.)

The question arises as to how control messages are to be addressed so that they can be sent to the appropriate nodes. For this purpose, each node could be associated with a distinguished LID which uniquely identifies the node throughout the network. Nodal conversion tables would have to be initialized appropriately. The same solution can be applied to loop 4, but, in addition, it is also possible to update LID entries in a node's ACT without knowing the nodal residence of the LID. By using the LID as the destination address in a control message, the message will be transported to the node containing the LID, and the updates required by the control message can then be made. It should be remarked, however, that this alternative approach on loop 4 may conflict with a desire to keep control channels separate from normal inter-process communication channels.

Although an LID may have at most one owner at a given node, it should be pointed out that an LID may not necessarily have the same owner at each node in the net. Whether or not an LID does have the same owner at each network node is a matter of host-level policy; for one thing, a determination of the consistency of all the ACT's is in large part an application dependent one, and for another, the inclusion of any such consistency checks within CIE firmware would complicate its operations significantly.

For a single process to establish a connection between two devices attached to the loop, it must own LIDs at both nodes. Ownership of LIDs at a single node permits the use of only one end of a connection. For example, suppose a process somewhere in the network commands the CIE attached to device one to mark some LID at the node as a legal local source address and another LID as a legal remote destination address for messages sent to device two. This action permits device one to send messages to device two, but unless the same LIDs at device two have been specified as legal remote source and local destination addresses, respectively, the messages will not be accepted by the CIE attached to device two.

Of course, two host-level processes may cooperatively establish a connection between network nodes if each owns LIDs at opposite ends of the connection. Consider the case of a process P1 running on host H1, and a process P2 running on Host H2. Suppose P1 owns LIDs 10 and 11 at H1, while P2 owns the same LIDs at H2. P1 marks LID 10 as both a legal local source and local destination address, and marks LID 11 as both a legal remote source and remote destination address; similarly, P2 marks LID 10 as a legal remote source/remote destination address and LID 11 as a legal local source/LOC destination address. In this way, messages may pass from H1 to H2 with source address equal to 10 and destination address equal to 11, while messages may pass from H2 to H1 with source address equal to 11 and destination address equal to 10.

The importance of the concept of LID ownership becomes obvious if we take a look at the centralized LID management scheme of the original ESM loops. In this approach, a single host is responsible for maintaining the conversion tables at each network node. Since no other host is programmed to manage these tables, no conflicts could possibly occur in the use of network LIDs (except through errors on the part of the centralized controller). Unfortunately, this is a rather inflexible scheme. The concept of LID ownership allows the management of LID space to be distributed among network nodes with the assurance that no host, or other suitably intelligent device, can modify or otherwise misuse the LID space of another. Each node can use the LIDs it owns (through its processes) to define its communication channels, and there is no possibility (except as noted below) of another node using these LIDS for its own purposes. This assurance is provided by the CIE's at each node - the CIE's ensure that nodes (processes) modify ACT's and nodal conversion tables only in ways that their capabilities permit. CIE controls are the only means of protecting against malicious or accidental misuse of network LIDs.

As suggested above, possible conflicts might occur in the use of LIDs if an LID does not have the same owner at each network node. A good example of this problem arises on Loop 4, where a process ID matrix is used at each node instead of the LID/functional address conversion tables of loops 1, 2, and 3. If a process P1 owns an LID at one node, while a proccess P2 owns the same LID at a different node, each process could choose to mark the LID as a legal local destination address (accompanied by appropriate changes to the process ID matrix) for their respective nodes. Thus, the node to receive a message addressed with this destination LID would be determined by the relative positions of the nodes and processors on the loop - an undesirable situation at best. A similar problem occurs on loops 1-3. Suppose process P1 owns LIDs at node N1, marking one as a legal local source address and another as a legal remote destination address; Process P2 owns the same LIDs at node N2, but marks them for use in communications with another node as a legal remote source address and a legal local destination address, respectively. If P1 specifies the residence of the LID at N1 as N2, N1 can send messages to N2 even though P2 had no intentions of establishing such a connection.

Such conflicts are avoidable only if ownership of LIDs is assigned to network processes in a controlled way. This is the function of host-level processes known as LID controllers. Theoretically, there could be any number of LID controllers in the network, each responsible for managing some portion of the network's LID space. They would do so, as suggested above, by assigning ownership of LIDs to network processes.

For most simulations, however, it would seem that the model need support only a single LID controller, which would manage the LID resource at every node in the network. Examples of this will be given later in this section. Only when there is concern for the reliability of the LID controller function, or its counterpart in the system to be modelled, would simulations involving more than one controller be necessary. Nevertheless, we will examine the problem of using multiple LID controllers as a means of discussing both the role of the LID controller, as well as the problems that arise when an access control function is to be distributed across several processes or nodes. The discussion is also important from the point of view that LID controllers have essentially the same relationship to owners, as owners have to LID users, the processes (or nodes) which actually use the LIDs for normal interprocess communications. The same considerations given below apply to owners and users, too.

There are a number of possible design choices with respect to the problem of associating multiple LID controllers with the LIDs they control. In the most general approach, the one admitting the greatest variety of simulation models, each LID at a given node would be paired with an LID controller in the node's access control table, and only this controller would have the right to determine the owner of the LID at the node. In this way, any number of LID controllers could manage designated portions of a node's LID space. Typically, an LID controller would manage the same set of LIDs at each network node, or some subset of network nodes. If a host wanted to establish a connection between itself and another device in the network (assuming it did not already own enough LIDs to do so), one of its processes would first have to contact a network LID controller. The controller, if the connection were authorized, could then assign ownership of one or more LIDs to the requesting process at the two nodes terminating the connection. Obviously, unless the controller has control of LIDs at both ends of the connection, more than one controller would have to be involved in the transaction.

A problem arises when an LID controller runs out of its LID resource, only to receive another request for an LID. Again, there is a variety of possible solutions. One approach is to have the LID controller contact another controller whose supply of LIDs has not yet been exhausted. The controller can allocate one or be demonstrated in section 3.5 of the report.

more (or none) of its LIDs to the requesting process (by making it the owner of the LIDs) as it finds appropriate. The decision would depend on the rights of the requesting process as given in access control tables maintained by the LID controller. When the process no longer required the LID, it would notify the original controller that the resource was no longer in use.

(Notice how a covert channel can be established between two processes if they share a common LID controller, and they are the only processes to use the LIDs it controls. Unlike the approach suggested above, assume in this case that the LID controller does not look for help when its supply of LIDs is exhausted, but instead merely notifies a requesting process of the condition, indicating that it should make another request later. To establish a covert channel, the two processes would first consume the LID resource by requesting ownership of all available LIDs. The sending process can signal a 1 or 0 by either returning an LID to the controller or doing nothing, respectively. The receiving process detects a 1 if, upon requesting an LID, it receives from the controller the only one currently available; otherwise, a 0 is signalled by the controller's denial of the request. Information could be exchanged in this way indefinitely. Clearly, care should be taken to program the LID controller so that there is no serious threat of information leakage over covert channels.)

When an LID request is made, the requester must somehow be identified to the LID controller, as must the location of the LID. For this purpose distinguished LIDs may be used to identify both the requester and the location of the LID. The use of distinguished LIDs implies that measures must be taken at network initialization time to ensure that appropriate host processes (e.g., LID controllers) be made aware of the correct association of network nodes with distinguished LIDs. Otherwise, even though a node uses its assigned distinguished LID, if some host process mistakenly associates a different node with the LID, host-level security controls are likely to break down completely.

The use of distinguished LIDs is only a partial solution to the problem. Ideally, in a network of "secure" processors, for example, processors could support a protocol which established the identity of their processes and associated these processes with unique LIDs. The processors would ensure that processes could use only their assigned LIDs in their messages. These LIDs could then be used to identify individual processes, instead of nodes, to an LID controller. (Alternatively, it would also be possible for hosts to support an interprocess communication scheme similar to that found in the ARPANET. In this case, multiple conversations are multiplexed over a single logical host-to-host connection. Each host is assigned a set of names which, like LIDs, serve as channel termination identifiers and are used to distinguish one conversation from another. Now, however, the channel identifiers are defined at the host, instead of the COMMNET, level.)

Of course, the precise role of the LID controller is really a function of network requirements. In some networks, for example, it may be sufficient for the LID controller to establish ownership of LIDs at network initialization time, so that ownership of LIDs would remain essentially unchanged throughout the lifetime of the network. LID controllers would intervene only in case of node failure, should CIEaccess control tables become damaged, or if, for example, the security-related capabilities of a processor, as recorded in its node's ACT, are to change when it runs a different mix of processes. In other networks, however, it would seem desirable to allow ownership of LIDs to fluctuate dynamically along with the changes in the communications needs of the network. In general, availability of the LID resource would probably be the determining factor in the decision to use LID controllers in other than those situations mentioned above. When there is enough to satisfy the needs of LID owners, LID controllers should have very little to do.

As mentioned earlier, the general approach outlined above provides for distributed management of the LID resource, and it is expected to have the advantages of higher reliability and throughput (though, in general, the LID controller function would hardly be a bottleneck) that are usually associated with a distributed processing approach. The controlled segmentation of LID space among LID controllers should permit them to manage the LID resource without interfering with each other. Although network LID controllers would generally be viewed as mutually cooperative managers of the LID resource, it is important that accidental errors be protected against by the CIE - LID controllers cannot necessarily be depended on for completely error-free operation.

When errors do occur, the CIE should signal a security montior, notifying him of the event. A logical ID identifying the monitor must be known to the CIE, and the CIE must be identified by a logical ID, unique within the network, in its communications with the security monitor. The CIE's LID should not be made available to any other process at the node.

There must also be a means of recovering from the loss of an LID controller, as, for example, when its supporting host goes down. In this case, there are two alternatives. The network can wait for the host to come back up again, but it will have to do without the services of an LID controller for perhaps an indefinite period of time. Alternatively, there could be some means of assigning a new LID controller to the LIDs once managed by the downed controller. This is the privilege of the node controller. The problem of reassigning LID space to another LID controller under these conditions is not necessarily an easy one, since processes requiring the services of the new LID controller will have to be notified of the change, and the new controller must be told of the current state of the LIDs it is to manage. Of course, the node controller can be used to redistribute LID space among its controllers under normal network conditions as well. As a final precaution against network failures, more than one node controller could be assigned to each node, so that if a node controller fails, another assigned to the same node can take its place. If all node controllers go down, manual restart procedures must be used.

It should be made clear that a process is node controller only for those nodes where it is so designated in an access control table. A node controller has access to all LID space at a given node, and reassigns LID space to another LID controller by changing the LID controller associated with the LID in the node's act.

The most general form of the model, then, can be summarized by examining a single entry in a nodal access control table, or ACT. There is one entry for each of the network LIDS, and each entry specifies the owner of the LID, an LID controller, and a capability map. The owner and associated controller are each designated by an LID - the LID to be used as a source address for control messages related to the LID entry. If either of these LIDs is zero, the LID is unavailable for use. The capability map consists of an LID attribute field and a process rights field. As described earlier, the attribute field consists of the source and destination fields (each two bits in length for the local source, remote source, local destination, and remote destination designations), and a delivery bit if desired. The process rights field specifies any rights that are to be associated with the process using the LID as a source address in its control messages to the node. The only example we have seen of this so far is the node controller right. Other rights may allow the process to modify a node's functional address (loops 1-3), or to change its write token address. All these rights are device-independent. A process may have, in addition, various device-dependent rights which, in the case of a CRT, for example, would permit the process to change the host that the CRT is logically connected to.

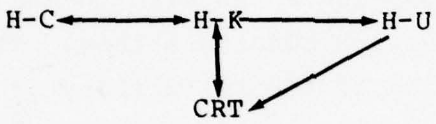
In addition, the ACT specifies both the LID used by the security monitor and the LID used by the CIE to identify itself in communications with the security monitor.

We will now give two brief examples of the model's use. Each will assume a less general form of the model by making use of only one LID controller. The LID controller will act as node controller throughout the network as well. Recall Cole's scheme for establishing a connection between two hosts in a network. Let us assume that the hosts belong to the same domain, and hence are associated with the same security controller, or SC. To establish a connection, one host must first send a request message to the SC (we ignore the case of simultaneous requests from both hosts). If the requesting host has authorized the connection, the SC will key the appropriate crypto devices so that communication can take place.

Though it does not rely on encryption for protection, our communications security model can support a similar connection protocol. To model Cole's approach, at network initialization time a single LID controller would assign to each security controller ownership of all LIDs at each node the SC controlled. Thus, no communications can take place without the cooperation of the security controller. When a host at one node requests a connection to another host in the same domain, the SC would update the ACT's at each node to allow the connection. The SC would ensure that no other nodes could use the same LIDs as were assigned to this connection.

As a second example, consider a small network of three hosts and several CRT displays which is to maintain a data base consisting of both classified and unclassified information. One host, H-U, maintains the unclassified data base, while another host, H-C, maintains the data base of classified information. The third host, H-K, performs a "kernel" function by preventing human operators from accessing the classified data base over CRT's unless they are authorized to do so.

H-K, assigned ownership of all network LIDs at network initialization time, could establish the following communications paths:



Note, first of all, that H-K will ensure that no classified information will ever be sent to the unclassified host H-U, where it could be transmitted to a CRT and displayed before an operator lacking the necessary clearance for classified information. Notice, too, that there is no communications path from H-C to the CRT. If there were, H-C could directly pass on classified information to an unauthorized user.

An operator at the CRT logs on to H-K, identifying himself by name and password, for example, H-K will then look up the operator's clearance level. If the operator is cleared for classified information, H-K will grant any requests for either classified or unclassified information; otherwise, only requests for unclassified information will be granted. H-K passes on any request for unclassified information to H-U which could then transmit the information directly to the terminal. Requests for classified information result in information transfers through the kernel, since no direct link can exist between H-C and the CRT.

The link from H-U to the CRT would also have to go if need-to-know categories were to be supported by the kernel. Otherwise, H-U could leak information to an operator who lacks sufficient need-to-know.

These two examples are only suggestive of what can be done with our communications security model. In general the model might possibly be used in two different ways - either as an approach to network security whose properties may be of interest in themselves, or as a tool for the simulation of alternative network architectures. Since the ESM loops permit any logical network topology to be simulated, and the model defines an approach to controlling the direction of information flow as well, the model obviously admits simulations concerned with COMMNET topology and its effect on denial of service considerations. The model would also be suitable for studies of the reliability of the COMMNET accesss control mechanism in its provision for multiple owners, LID controllers, and node controllers. In addition, much can be done within the constraints of the simulation model to study the role of owners and controllers in a network; the procedures and protocols for establishing connections, identifying processes to remote hosts, or recovering from failures in COMMNET access control; and more generally, the requirements for controlling a communications subnet in a secure fashion. The interaction of the host-level security policies with communications security policies can also be studied.

The model, however, has obvious inadequancies - both as a general protection scheme and as a tool for simulations. The model was never intended, for example, to provide process-level access control to the COMMNET. If a host may transmit down any one of several channels, as determined by the node's CIE access control table, any process on the host has access to those channels. Such controls, if desired, must be included in host-level software.

Less obvious, perhaps, is the following example. Suppose a node A used LIDs 5 and 6, each marked as both legal local source and remote destination LID. To receive messages from A, node B uses the same LIDs, but, in this case, they are marked as both remote source and local destination LIDs. Originally, LID 5 at node A was paired with LID 5 at node B to form one simplex channel, while LID 6 was used at both nodes to form another. With both channels present, however, node A may now send messages to B with 5 as the source address and 6 as the destination address. This would allow, for example, the process associated with LID 5 at node A to communicate with the process associated with LID 6 at Node B. In terms of the model, this apparent "misuse" of allocated channels would not be viewed as a security violation since the nodes clearly have the right to communicate with each other. A higher (Host) level protocol is required to avoid this problem.

As another example of the model's limitations as a general protection scheme, suppose that two hosts wished to establish a communications channel between them. It would be desirable if the two hosts could be sure that the connection, once made, did in fact terminate at the two hosts, so that each could be certain of the identity of the host, or device, at the opposite end of the connection. If the two hosts cooperate in making the connection, there are no problems. But if neither host quite trusts the other, each believing that the other might substitute a different host at its end of the connection, then the model provides no mechanism by which the connection can be made, except by way of a higher authority (a host-level process owning LIDs at both nodes) which could establish the connection for them. Note, however, that this limitation does not prevent such connection establishment protocols from being studied on the ESM loops. Host software can easily be developed to simulate "secure" hosts capable of cooperatively establishing such communications channels between them.

The usefulness of the model as a simulation tool is limited as well. Its application, obviously, is restricted to studies of access control with respect to the communications medium. Access control to other objects plays no part in the model at all. Moreover, the model ignores several aspects of a communications subnet that can have a decided impact on network security. For example, the physical topology/interconnection scheme of the subnet (e.g., loop, star, circuit switched, message switched, etc.) may make a computer network more or less vulnerable to attacks directed at denying service to its users. COMMNET-Level protocols may influence network security as well. Nor is the model concerned with network security problems related to encrpyption. These problems and others can be studied in simulations on the ESM loops using host-level software, but the model itself is simply not general enough to allow for these variations in network architecture. In the next section, we will discuss several important issues which cannot effectively be studied within the constraints of the simulation model, and consider once again other problems of network security only briefly mentioned earlier in the report.

3.6 Security Characteristics of Alternative Network Architectures

This section takes a closer look at some of the parameters of a network architecture that affect its security. The emphasis here is on those architectural features which have not yet been given sufficient attention in the report, or cannot be adequately studied in terms of the communications security simulation model. Again, there is no attempt here at completeness, only a desire to present some critical design issues in the area of computer network security.

To begin with, very little has been said about the effects of the COMMNET's interconnection scheme (loop, star, message-switched, circuit switched, etc.) on the overall security of a computer network. The following discussion on this subject is taken largely from (Col 75).

If it is assumed that the communications subnet operates on encrypted data, the basic security threat to the network is one of denial of service. With this in mind, Cole examines seven different interconnection schemes; point-to-point, circuit switched, tree, star, multiply-connected, loop, and radio broadcast.

Point-to-point network. "A seemingly straightforward approach to controlling access between network entities is to directly interconnect all those devices authorized to communicate with each other, such that only those connections would exist in the net. If a given entity such as a host would change its security level during the day, an appropriate portion of its links would be enabled or disabled, giving some ability to adapt to change.

"Several problems plague this scheme. In all but the smallest nets, the number of interconnection combinations quickly gets out of hand, since the number of meaningful connections tends to be a sizeable portion of the N(N-1) different possible links connecting N to entities. Also, implied connections via possible n'th party access tend to circumvent the careful isolation of the different entities, unless a hierarchical authorization scheme exists, which in itself is not necessarily proper security. Other all or nothing aspects to such an arrangement tend to violate our concepts of how network access should be determined and controlled. Therefore, the dedicated connection net represents one class of network structure, which is an interesting point on the spectrum of possibilities, but one that is too extreme for any general utility." (Col 75)

Circuit-switched network. "The best example of a circuit-switched net is the direct dial telephone system, which of course can be utilized for data communications as well as voice. The principal problems in such usage are the limited bandwidth and the time required to establish the connection, while the primary advantages are its widespread existence and availability...

"The direct distance dial net is particularly inefficient for the interactive user, who typically can never utilize the full line capabilities and multiplexing of a dial connection is feasible only under conditions which tend to contradict the availability advantages of the direct dial net. A combination of direct dial and multiplexed point-to-point lines is often utilized, but borders on other combination nets such as a message-switched net with direct dial access.

"The security-related aspects of the direct dial net are largely related to its impact on the cryptographic equipment, and in particular, whether multiplexed cyrpto devices are economically advantageous. For example, if individual direct dial lines are brought to a host computer, they would then have to be multiplexed prior to entering the crypto device. Much of the flexibility of addressed multiplexing, e.g., by the use of message headers, is therefore lost, and many of the physical port constraints begin to show up on the design of the multiplexed crypto devices. Handling of the large number of input lines, connectors, etc., may also grow beyond expectations for such usage."

(A minor security advantage of the direct dial net is) "The difficulty that an enemy would have in performing any meaningful traffic analysis. There is also a large (apparent) redundancy in the direct dial net, but there are probably a number of sensitive points which would be very vulnerable to sabotage and would thereby sever a large portion of the user community from the net. In addition, malicious users might tie-up all of the input ports, thereby denying service to legitimate users." (Col 75)

Tree-structured nets (message-switched). "The tree structure is occasionally utilized for networks when its relatively low line cost, and hierarchical organization match the needs of the network community, and when its high vulnerability to loss of any link is acceptable (or correctable by back-up methods). (The tree net has limited utility in applications) which require high availability of basically non-hierarchical resources." (Col 75) Nevertheless, it may be well suited to the hierarchical control structure associated with the system control application.

Star Nets. "The star topology is also very vulnerable to loss of components, particularly the central switch, and to a lesser extent, to any link since that loss would sever one entity from the net. Line costs would also be high if the network entities are separated by inter-city distances, and the operational performance can degrade rapidly when a large number of small messages must be handled concurrently (i.e., switch saturation). There are some minor positive factors as well, such as the convenient spot for monitoring operations, namely the central node. However, since it is vulnerable to overload, adequate monitoring may not be feasible.

"Denial of service is the greatest security threat of the star network, particularly due to its exceptional vulnerability to the loss of components or message flooding as discussed above." (Col 75)

Multiply-connected message-switched nets. "The reliability/ availability disadvantages of star and tree nets can be overcome by adding redundant links between the nodes. The particular structure of the net can then become independent of any predefined topology, and instead, can be based on expected traffic loads and geographical locations. The ARPA network is the prime example of this type of network...

"The major security related advantage of the multiply-connected message-switched net is its high resistance to errors and/or malicious damage. This flexibility is, at the same time, its only apparent security disadvantage since complication tends to breed exploitable combinations of events and circumstances. This subjective observation is not an indictment against message-switching; it is merely a word of caution in the usage of what appears to be the best available data communication technology available." (Col 75)

Loop (ring) networks. "One of the most attractive aspects of a loop net is based on the fact that each entity on the ring sees every message as it goes by, and therefore, one can address messages to a given process (instead of to a physical processor). The only requirement is that each interface be able to match process names (from message addressing) with a list of current processes which it contains. Other advantages are based on the expected low cost of the interfaces between the devices and the net, and the simple communications technology which utilizes only digital devices (similar to the telephone T1 carrier)."

"The single loop is inherently vulnerable to the loss of a line segment. Although back-up paths can be added, the increase in complexity and added line costs tend to detract from the attractiveness of the loop except for well controlled, local environments. Therefore, the loop net would seem to be an appropriate candidate for a local subnet, but not for the global subnet to interconnect such subnets."

"Other security-related aspects include some increased vulnerability to traffic analysis, since all messages go by any given spot on the ring. However, message headers could be encrypted (with a common key) to avoid this problem." (Col 75) on a more positive note, this characteristic of the loop architecture allows a "security monitor" to be inserted anywhere in the loop for the purpose of monitoring network traffic. Cole also notes the difficulty of implementing a priority override scheme on a loop. On the ESM loops, for example, a node must wait to receive a write toekn before sending off any of its messages. There is no obvious way of allowing a node to preempt the loop resource for the purpose of transmitting a high priority message through the net ahead of other traffic. Other loop protocols have similar difficulties.

Radio Broadcast Nets. Among the possible drawbacks to the use of radio broadcast nets, Cole mentions: (1) the lack of available frequency spectrum, (2) geographical coverage problems, (3) ease of tapping the communications media, and (4) denial of service threats by jamming the radio net.

Cole neglects the use of a bus as a network interconnection scheme. Nevertheless, ETHERNET (MB 75) has demonstrated the effectiveness of a bus architecture for local networks. Though a break in the bus would divide the network in two (and, in addition, gives rise to interference problems due to signal reflections at the breakage point), the failure of any node affects the communications of only a single device since the communications facility provided by the ETHERNET is a passive one. Once access to the bus is permitted, the bus allows maximum wire-speed transmission with minimal delay. Only one transmission can take place at any time, however.

A network's interconnection scheme should be examined for its impact on network security both from the point of view of: (1) its "communications topology" and (2) the connection protocol used in the COMMNET to control the transmission and receipt of network messages. For example, a loop defines a network in which all nodes are physically connected to each of two other nodes in the network; though each node directly transmits information to only one of its neighbors, information can be passed from one node to the next so that each node is logically connected to every other node. Unless redundant links are used, however, a single line fault or node failure will severely affect communications on the loop. In addition, a timing node is generally used to ensure that an integral number of bits is always present on the loop,, so the loop is only as reliable as its timing node. All of these considerations are related to a network's communications topology. Error and flow control are included under (2). If transmission errors go undetected, security-related information contained in network messages may be altered, possibly resulting in a security compromise. And we have already noted, for example, that a simple write token protocol may not be appropriate in a loop network when high priority traffic must be given more rapid access to the net.

These aspects of a network's architecture are ignored entirely by our communications security model - the model defines a higher level of network design. The ESM loops, and the simulation model, do admit experiments concerned with the logical interconnection scheme of a network (as defined by the presence or absence of connections between network nodes), but there is no immediate way to take into account the effect on network security of redundant physical links, of circuit-switched connections, of alternative strategies for flow control, etc. These variations in network architecture must be simulated in host software, or additional CIE firmware.

Another important parameter of a network architecture, mentioned in section 3.4 of the report, is the nature/location of its security controls. As defined by our communications security model, for example, the CIE's on the ESM loops play a significant, but minimal, role in COMMNET security. The communications processors of Farber and Larson (FL 75), on the other hand, are much more complex and, in fact, are intended to overcome altogether different problems of network security. In their design, "reasonably secure" hosts interface to a loop network via communications processors. According to Farber, the communications processor was chosen as the major component in network security because of "the lack of a uniform environment within hosts at different sites, and the increased protection (that) a separate, autonomous component, the communications processor, creates."

The loop transports messages addressed by process name. "The security scheme...is based on the idea that if we are capable of dynamically changing the names of intercommunicating processes, then if the names are changed frequently enough, an observer looking at messages passing over the communication subsystem will be unable to tell who is communicating with whom...in addition, it is difficult for an intruder to tell by what name a transmitting process will be addressed in the return message; thus, it will be extremely difficult for the intruder to create false messages." The processors, then, were not designed to restrict host-to-host communications, as is the case with Cole's scheme, for example, but instead, to protect against outsiders monitoring network traffic. What is proposed, really, is an alternative to the encryption of message headers.

The details of Farber's scheme are not of real importance here - the brief discussion above should, however, illustrate the range of variations possible in COMMNET level security controls, especially when network security requirements differ. Once again, Farber's scheme departs radically from any our own communications security model would support. As another example, the BDS nodes of loop 4 could be used to support much more sophisticated security controls than those forming the basis of the communications security model discussed in section 3.5 of the report. In particular, they could be programmed to provide controlled terminal access to network hosts. Instead of logging on to a host directly, a user would first identify himself to a BDS and request access to some network resource. The BDS would then connect the user to the appropriate host if the request were authorized. In this way, a network host is relieved of the burden of interacting with users who intend to access a different network host.

The nature of host-level security controls may also vary considerably. Our COMMSEC model makes no general assumptions about the behavior of host processors. Other network designs, however, require host processors to satisfy a number of security-related requirements. In a network of multi-level security (MLS) hosts, for example, each host must provide for simultaneous, controlled access by users/processes of different clearance levels and the need-to-know to information of different classifications and categories. This can be accomplished by way of a processor "kernel", or reference monitor, which is automatically invoked whenever information is accessed to enforce MLS authorization requirements. Constrast this "secure processor" approach with that previously taken by the military whereby information of only a single classification level was allowed on a host system at any given time, and changes to a lower level, for example, involved purging the system of any sensitive information it contained. Verification of the correct operation of a "security kernel" is a difficult task, but the flexibility and expected cost-effectiveness of the approach seems sufficient justification for its use.

Host-level security requirements must, of course, be supported within the communications network. The COMMNET must ensure, for example, that top-secret files or messages are kept separate from those of different classifications. When priorities are associated with network traffic, response and preemption requirements must also be satisfied by the COMMNET. Obviously, COMMNET security requirements are not formulated in a vacuum.

Earlier we saw another example of how constraints on host behavior could lead to a more flexible network design by providing process-level, rather than host-level, protection guarantees. In this way, for example, if a CIE supporting our COMMSEC model received a message from an LID controller, there would be the assurance that the message was actually sent by the LID controller, and not by some other process contained in the same host. It would also be possible to make guarantees concerning the transmission and receipt of interprocess messages, even though the processes are located at different network nodes. The problem, of course, is one of specifying the necessary enforceable constraints on host behavior which would permit such guarantees to be made.

The most general network design discussed in this report, that presented in (USAS 76), also proposes a "secure processor" approach. It defines an abstract machine which must be supported at every node in the network. The primitives of this machine allow a network process at one host to allocate its resources to another process at a different host, and the allocating process need only know the "name" of the process to receive the resource. Furthermore, the allocating process may preempt the resource from the remote process should the process refuse to return it. The abstract machine supporting the remote process will return the resource at the request of the allocating process. Again, to prove that an implementation of this abstract machine is correct would be a difficult task, but it need be done only once if hosts and communications processors are essentially identical throughout a network. Of course, whether or not these and other protection features would be required in a SYSCON network is not yet known. The USAS model does suggest, however, what can be done in the area of security when appropriate constraints are placed on the behavior of host processors.

Finally, the question of just what security controls should exist at a given level of the network design depends critically on what controls are assumed to exist at other levels. As noted earlier in this section, for example, Farber and Larson's choice of a communications processor as the focal point of network security was based, in their words, on "the lack of a uniform environment within hosts at different sites, and the increased protection a separate, autonomous component, the communications processor, creates." (FL 75) The same arguments apply to our communications security model. The USAS design, however proposes to establish just such a uniform environment at host sites in order to facilitate inter-nodal interactions, and to provide processlevel protection on a network-wide basis. Hosts can now be depended on to enforce process-level communications constraints, whatever they might be. Additional security controls are no longer needed in a communications processor to protect against malicious host behavior.

We complete this section with a brief discussion of crypto controls and their contribution to network security, considering first the problems of traffic analysis and spoofing before going on to present other issues related to data encryption.

Traffic Analysis. In a message-switching environment, for example, switches must have access to the routing information contained in the message headers. Unfortunately, if routing information is sent in the clear, outsiders tapping the communications lines will have access to the same information. Farber and Larson (FL 75) proposed one solution to this problem, as we have seen. More typically, however, link encryption is used to protect against traffic analysis, so that message headers are passed in encrypted form from one node to the next, decrypted within the node for routing purposes, then encrypted once more before transmission to the next node.

Spoofing. Cole (Col 75) states: "In essentially every network, it is possible for some person to tap into the communication facility in such a manner that he can modify otherwise legitimate messages or can create extraneous messages (by playing back recorded messages, possibly modified, for example). Such actions are referred to as spoofing, and are performed with the intent of either: (1) causing improper actions to take place, (2) causing confusion at the host sites or in a network control center, or (3) degrading service in some major way such as by creating erroneous routing table updates in a message-switched network. Spoofing threats can be countered by (1) detecting modified messages by use of error checks on the clear text, (2) detecting the replaying of legitimate messages by the use of encrypted sequence numbers or time stamps, and (3) discarding any messages that do not meet these checks.

Data Encryption. The issues of interest here concern the use of link versus end-to-end encryption, and the integration of encryption techniques into a network composed of smaller subnets whose physical controls are such as to eliminate any need for encryption within the subnets themselves. The choice of end-to-end encryption versus the simpler, and less expensive link encryption approch to data security largely depends on whether or not the switches on a message path can be "trusted" with clear message text. If not, end-to-end encryption of the text, from source to destination switch, is required.

Our own communications security model does not depend on encryption to protect information passing through the network. It assumes instead that physical security is adequate enough to prevent outsiders from tapping the communications lines. Furthermore, it assumes that the COMMNET interface can be trusted with clear message text. Under these conditions, if loops, for example, are used for the smaller, local subnets, link encryption between gateway nodes should provide adequate protection against outsiders. Of course, in some networks end-to-end encryption may be desirable in that it can provide additional confidence that the COMMNET cannot misuse the message text in any way. Although an expensive solution, and one whose complexity makes certification of the protection mechanism itself non-trivial, an end-to-end (source to destination gateway) encryption scheme much like that described by Cole (Col 75) might be the only recourse when an extra measure of security assurance is required.

3.7 ESMD Loop 4 Security Monitor Demonstration

Node H (Node Designator 19) of ESMD Loop 4 will be connected to a general purpose processor (e.g., PDP11/70) which could be used as a security monitor. An experiment/demonstration which could be performed to illustrate the use of the security monitor with respect to the Loop 4 automatic loop-back feature which can remove a security violating node from the system is described below:

i) The security monitor processor is used to monitor a terminal-host dialogue. It sends control packets to the loop 4 B776 processor node (Node Designator 16) and the loop 4 CRT terminal node (Node Designator 18) such that their primary read addresses are read non-destructive. This would involve modifying the address comparison memory chip so that read address 4 is read non-destructive for the B776 (HSTC) node and read address 7 is read non-destructive for the CRT node (CRT 18).

ii) The security monitor modifies the address comparison memory of its connected node such that it reads non-destructive both addresses 4 and 7.

iii) The security monitor displays the CRT-host dialogue on one of its local terminals.

iv) The monitor checks the validity of the password entered on CRT18.

v) When the password fails, control packets are sent out on the loop in order to remove CRT18 from the system. This is done by sending a control packet to node 17 to do a backup line switch (BKLNSW), and a control packet to node 19 to do a primary line switch (PRLNSW).

3.8 Conclusion

We have reviewed the general network security problem. In addition a communications security simulation model was described whose purpose was (1) to illustrate one approach to COMMNET access control and (2) to serve as a basis for the simulation of a small family of network architectures. Given this general background to the network security problem, what is needed now is to determine the special requirements of a network for system control - in particular, those related to network security. Once these requirements are well understood, the simulation model should prove useful in evaluating different network architectures intended to support SYSCON security requirements. There is, of course, no substitute for designing and testing the actual system. Appropriate simulation experiments should, however, help pinpoint many design flaws before they are committed to a final implementation.

(Alv 64) Von Alven, William H., Reliability Engineering, Arinc Research Corporation, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1964.

(Bl 73) Bell, D. E. and L. J. Lapadula, Secure Computing Systems: A Mathematical Model, MITRE Corporation, Beford Mass., 1973.

(Col 75) Cole, G. D., Design Alternatives for Computing Network Security, SDC, 1973.

(Cos 75) Cosell, B. P., Johnson, P. R., Malman, J. H., Scahntz, R.E., Sussman, J., Thomas, R. H., and D. C. Walden, "An Operational System for Computing Resource Sharing," Proceedings of the Fifth Symposium on Operating Systems Principles, November 1975.

(CS 76) Chandersekaran, C.S. and K. S. Shankar, "On Virtual Machine Integrity," IBM Systems Journal, August 1976.

(Dat 76) Date, C.J., An Introduction to Data Base Systems, Addison-Wesley, January 1976.

(Dend 76) Denning, Dorothy E., "A Lattice Model of Secure Information Flow," Communications of the ACM, Volume 19, No. 5, May 1976.

(Denp 76) Denning, P. J., "Sacrificing the Calf of Flexibility on the Altar of Reliability," Proceedings of the Second International Conference on Software Engineering, October, 1976.

(FL 75) Farber, David J. and Kenneth C. Larson, "Network Security via Dynamic Process Renaming," Proceedings of the Fourth Data Communications Symposium, October 1975.

(Lev 75) Levin, R., Cohen, E., Corwin, W., Pollack, F. and W. Wulf, "Policy/Mechanism Separation in Hydra," Proceedings of the Fifth Symposium on Operating Systems Principles, November 1975.

(MB 75) Metcalfe, Robert M. and David R. Boggs, ETHERNET: Distributed Packet Switching for Local Computer Networks, Xerox Palo Alto Research Center, CSL 75-7, November 1975.

(Neu 75) Neumann, P. G., Robinson, L., Levitt, K. N., Boyer, R. S., and A. R. Saxena, A Provably Secure Operating System, SRI Project 2581, Stanford Research Institute, Menlo Park, Calif., June 1975.

(RK 76) Rammamoorthy and Krishnarao, "The Design Issues in Distributed Computer Systems," Distributed Processing, INFOTECH, 1976.

(Rot 74) Rotenberg, Leo J., Making Computers Keep Secrets, Ph.D. Thesis, MIT, MAC TR-115, February 1974.

(Sch 74) Schutzer, D. et al, DCA System Control Concept Fomulation, Defense Communications Engineering Center, Reston, VA February 1974.

(Schr 72) Schroeder, M.D., Cooperation of Mutually Suspicious Sub-Systems in a Computer Utility, Ph.D. Thesis, MIT, MAC TR-104, September 1972.

(Sha 76) Shankar, K.S., The Total Computer Security Problem with Solutions, Approaches, and Examples, Internal Working Paper, Burroughs Corporation, February 1976.

(Usas 76) Unified Software Architecture Study, DCA 100-75-C-0073, December 1976.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER 14 64297-VOL-1	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER 9
4. TITLE (and Subtitle) 6 ~~Final Report for the~~ Exploratory System Control Model Development (ESMD), ~~Vol I~~ Volume I. Sections 1-3.		5. ~~TYPE OF REPORT~~ & PERIOD COVERED Final Report Jul 77 - Jan 78.
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) 18 SBIE 19 AD-E100 144		8. CONTRACT ~~OR GRANT~~ NUMBER(s) 15 DCA 100-76-C-0081
9. PERFORMING ORGANIZATION NAME AND ADDRESS Burroughs Corporation Federal and Special Systems Group Paoli, PA 19301		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Task 15203 P.E. 33126
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Communications Engineering Center 1860 Wiehle Avenue Reston, VA 22090		12. ~~REPORT~~ DATE 11 Jan 78
		13. ~~NUMBER~~ OF PAGES 149
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same as 11 12 153p.		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same as 16

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

loop, ring, system control, Defense Communications

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report covers modeling for System Control distributed data base for fail-soft architecture and for secure system architecture.

070 040

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)